A Characterization of the Use of the UNIX C Shell[1]

M.S. Report - Plan II

Rita K. Hanson

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California
Berkeley, California

Abstract

The Unix C Shell was modified to record users' input, and data was gathered from a DEC VAX 11-780 running UNIX 4.2 BSD and UNIX 4.3 BSD. This data from the Unix C Shell was combined with accounting data and post-processed to create clean formatted data for easy analysis. The resulting post-processed data was analyzed to determine users' usage patterns.

The post-processed data is useful because each record contains a large quantity of information, including the entire command line. Previous studies have recorded command frequency patterns, but have not recorded entire command lines. In addition, the formatted post-processed data will make future analysis relatively easy.

Analysis results will provide useful data for designers of future command interfaces, and they may suggest improvements to the Unix C Shell. The analyses completed to date have shown that a few commands account for the vast majority of the data. In most cases, usage patterns are relatively simple. The average user only uses about one-fifth of all commands, and the vast majority of all command lines contain only one command. Users, however, often alter their environment by using aliases.

In may cases, analysis produced predictable results. Users often execute commands in the background that use a relatively large amount of CPU time. Certain complex features are very seldom used. Programmers and graduate students tend to have more complex usage patterns than secretaries.

One surprising result was that almost 30 per cent of all the data was generated by system command files. The *daemon* program generated 28.4% of all commands. This result, however, is probably a unique characteristic of the machine used for data collection, and is not characteristic of typical C Shell use.

[1]The material presented here is based on research supported in part by the National Science Foundation under grant DCR-8202591, and by the Defense Advance Research Projects Agency (DoD), under Arpa Order No. 4871 (Monitored by the Naval Electronic Systems Command under Contract No. N00039-84-C-0089) and Arpa Order No. 4031 (Monitored by the Naval Electronic Systems Command under Contract No. N00039-84-C-0235). Partial support was also provided by Digital Equipment Corporation and a MICRO grant from the University of California.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 1985		**00-00-1985 to 00-00-1985**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Characterization of the Use of the UNIX C Shell	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Unix C Shell was modified to record users' input, and data was gathered from a DEC VAX 11-780 running UNIX 4.2 BSD and UNIX 4.3 BSD. This data from the Unix C Shell was combined with accounting data and post-processed to create clean formatted data for easy analysis. The resulting post-processed data was analyzed to determine users' usage patterns. The post-processed data is useful because each record contains a large quantity of information, including the entire command line. Previous studies have recorded command frequency patterns, but have not recorded entire command lines. In addition, the formatted post-processed data will make future analysis relatively easy. Analysis results will provide useful data for designers of future command interfaces, and they may suggest improvements to the Unix C Shell. The analyses completed to date have shown that a few commands account for the vast majority of the data. In most cases, usage patterns are relatively simple. The average user only uses about one-fifth of all commands, and the vast majority of all command lines contain only one command. Users, however, often alter their environment by using aliases. In many cases, analysis produced predictable results. Users often execute commands in the background that use a relatively large amount of CPU time. Certain complex features are very seldom used. Programmers and graduate students tend to have more complex usage patterns than secretaries. One surprising result was that almost 30 per cent of all the data was generated by system command files. The daemon program generated 28.4% of all commands. This result, however, is probably a unique characteristic of the machine used for data collection, and is not characteristic of typical C Shell use.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**94**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Table of Contents

1. Introduction

As hardware costs decrease and salaries increase, programmer and staff productivity become more and more important. Previous studies, e.g. [Lambert84] and [Thadhani81], have shown that decreased system response time has increased programmer productivity. Good computer services can increase programmer productivity, also, as shown in [Thandani84]. Sophisticated user-friendly interfaces to the computer are also likely to increase programmer productivity. In addition, good user interfaces will make computers easier for the naive or occasional users to use, increasing their job productivity.

Studying usage patterns of present user interfaces will help in the design of future user interfaces. Data showing which currently-available commands and features are most frequently used will indicate which commands and features should be most readily available and easy-to-use in future user interfaces. For example, the C Shell, a command interpreter for Unix, includes many popular features of other command interpreters. It has become quite popular among Unix users. (Unix users may choose to use the C Shell, or the original Unix command interpreter, the Bourne shell.) Formal studies of the use of the Unix C Shell may suggest further improvements to it and other command interpreters. Differences in various users' usage patterns are also important. Future user interfaces should be easy for both experienced programmers and occasional users to use. If different types of users have different usage patterns, then future user interfaces will have to adapt to these varying patterns. For example, less experienced users need help facilities and easy-to-use commands, while more experienced users are less interested in these features.

With the above goals in mind, this paper characterizes the use of the C Shell and describes how the results were obtained. The data was collected by modifying the C Shell to record command lines as they were typed by users. This data was combined with accounting data that was already being generated. The results were post-processed to create clean easy-to-analyze data. The post-processed data was then analyzed to characterize C Shell usage patterns.

1.1. Goals and Requirements

My goals were to produce data in a format that would make further analysis easy, to minimize the effect on the system during data collection, and to produce a comprehensive study of usage patterns for a command interpreter.

1.2. Previous Work

Kraut and Hanson in two joint papers, [Hanson84] and [Kraut 83], provide some frequency statistics for twenty Unix commands and provide correlations between commands executed sequentially or within a few commands of each other. Penniman [Penniman84] showed that a small set of usage patterns account for the bulk of activity. He showed that frequent, moderate, and infrequent users have significantly different usage patterns. Rosson [Rosson84] showed that the type of work that users did had a strong affect on their use of complex features. He also suggested that programming experience is helpful in using these more complex features. Further discussion of previous work is in Section 4.2.7.

The above results did not include the detailed statistics that I wanted. Extensive study of the use of aliases, history substitutions, input/output redirection, and pipes has not been done. This papers presents detailed results that show the use of these and other C Shell features.

1.3. Outline of Paper

In Section Two, design issues will be discussed. Implementation issues will be discussed in Section Three. Analysis results are presented in Section Four. The performance of the data collection program is discussed in Section Five, and Section Six

lists lessons I learned during design and implementation. Appendix A gives a brief description of those features of the Unix C Shell that were analyzed. Appendices B, C, D, and E describe the formats of the data records. Appendices F and G give tables that were used to derive information in the body of the report.

2. Design Issues

This section discusses issues encountered during the design of the data collection method, the data post-processor, and the data analyzer. Issues encountered during the implementation of the design are discussed in the succeeding section.

2.1. Choices for Data Collection

2.1.1. C Shell

The C Shell *(csh)* is a command interpreter for Unix. It was chosen for study because it was used by the vast majority of the users on the machines that were easily available for data collection. Unix users may also use the Bourne Shell, but very few users on the machines available for data collection use the Bourne Shell. The C Shell's features include job control, input/output redirection, pipes, history substitutions, and alias substitutions. For more information on these features see Appendix A, the *csh* manual entry in [Computer84a], or "An Introduction to the C Shell" in [Computer84b]. The C Shell should not be confused with the Bourne Shell *(sh)*, another command interpreter for Unix, which was not studied. For more information on the Bourne Shell, consult the *sh* manual entry in [Computer84a], "An Introduction to the Unix Shell" in [Computer84b], or [Bourne78].

2.1.2. Gathering Data

I considered three alternatives for gathering data about C Shell usage. The first alternative was simply to use existing Unix facilities. For each command, Unix generates a record in an accounting file. Each record includes the command name, a user identifier, the beginning time, and resource usage. Since this data was already being generated, this offered an easy method for data collection.

A second alternative was to gather the data directly from the C Shell, as it was read from the terminal. While this method is not as simple as using the accounting data, it has many advantages. This method allows collection of the entire command line. I was interested in not only which commands users invoke, but how they are invoked. For example, pipe usage, input/output redirection usage, the number of commands per command line, and whether the command was executed in the background or the foreground show how users invoke commands. This information is available in the command line, but it is not available in the accounting file. Another advantage of collecting the data from the C Shell is that the C Shell performs history and alias substitutions. So, I can record the command line both before and after history and alias substitutions are made. (See Appendix A for a brief explanation of the history mechanism and alias substitutions.) Also, 59 commands that are available to the user are executed directly within the C Shell. (See Appendix A for a brief explanation of built-in commands.) These commands do not appear at all in the accounting file. My results show that 24 of the 50 most frequently used commands are commands built directly into the C Shell. Consequently, using the accounting file would have prevented gathering data for these commands.

I chose a third alternative, which was a combination of the first two alternatives. Data was gathered directly from the C Shell, and then combined with data from the accounting file. While this alternative was the most difficult of the three alternatives to implement, it also provided the most data. This alternative provided both the command line data from the C Shell and the resource usage information in the accounting file.

2.1.3. Environment for Data Collection

Data was gathered over a two week period (July 9 - July 22) on a VAX 11-780 (ucbarpa) running a version of BSD Unix. Since 4.3 BSD Unix was under development at the time, the operating system contained parts of 4.2 BSD Unix and parts of 4.3 BSD Unix. Most users were faculty members, graduate students, and programming staff engaged in research. A few secretaries were also using the machine. The load average (the average number of runnable processes) on the machine varied greatly, but it was typically two to five during peak hours (10 a.m. to 5 p.m.), with occasional bursts of activity causing higher load averages. During less critical hours, the load average was typically around one or two, and during the night (midnight to 8 a.m.), it was typically less than one-half.

2.2. The Structure of the C Shell Tracer

My design of the data collection, post-processing, and analysis programs required four steps. During the first step data is collected from the C Shell. The second and most complex step involves merging the data collected from the C Shell with the accounting data. During the third step the data is post-processed to allow easy analysis. Finally, in the fourth step the data is analyzed.

2.2.1. Collecting Data from the C Shell

I wished to record the command line and whether the command line was typed by the user at his terminal, or whether it was part of a command file. If it was part of a command file, then whether the command file was a .login, .cshrc, .logout, or some other type of command file was also recorded. (See Appendix A for an explanation of command files.) Since I wanted to study which features of the C Shell were being used, not just which commands, the entire command line was recorded. Also, since I wished to study the use of the history mechanism and alias substitutions, I recorded the command line both before and after history substitutions and aliasing had taken place.

2.2.2. Combining the Trace Data and the Accounting Data

After the data from the C Shell has been recorded, it must be combined with the data in the accounting file. The C Shell data contains command lines, and users may type multiple commands on the same command line. The accounting file, however, contains only commands. So, each command line in the C Shell data must be parsed to obtain the commands. The commands from the C Shell data file can be matched to the commands in the accounting file by examining the time, user identifier and the command itself in each file.

A few complications arise in this design. The first complication is that some commands are executed directly within the C Shell and, as a result, do not appear in the accounting file. This problem can be ignored by not attempting to match these commands. The second complication is that some commands in the C Shell data are actually from Bourne Shell *(sh)* scripts, and correspond to multiple commands in the accounting file. (See Appendix A for a brief explanation of scripts.) To process the scripts properly would require tracing the Bourne Shell *(sh)*, along with the C Shell *(csh)*, and then combining the data. This would add another step to the processing and introduce additional complexity. Since very few users on the machine I was tracing used the Bourne Shell, I guessed that Bourne Shell scripts account for only a small percentage of all commands. As a result, I decided not to trace the Bourne Shell. So, instead of attempting to find accounting commands for all scripts, I only attempt to find accounting commands for C Shell scripts.

A third complication results from commands that invoke other commands. Any command or program may execute another command or program, without using the C Shell. These programs executed by other programs generate accounting records. Since

these program-generated commands do not correspond to anything the user typed in the C Shell, I cannot determine which C Shell command line to attribute the accounting commands to. Except for five popular programs handled by special-case code (*cc*, *lint*, *make*, *rdist*, and *sccs*), I did not attempt to match accounting records for program-generated commands.

The purposes of these five commands are not relevant to this paper, so only brief explanations of their purposes will be given. If, however, the reader desires additional information, consult the appropriate reference. *Cc* is the Unix C language compiler. For more information on the C programming language, consult [Kernighan78]. *Lint* is a program that examines C source programs and locates bugs and potential portability problems, explained more thoroughly in "Lint, A C Program Checker" in [Computer84b]. *Make* ensures that the proper source files are compiled when necessary, eliminating the need for the programmer to manually keep track of which source files need to be re-compiled. A more detailed explanation is given in "Make - A Program for Maintaining Computer Programs" in [Computer84b]. *Sccs* manages changes to programs and different versions of programs. "An Introduction to the Source Code Control System" in [Computer84b] provides more information. *Rdist* keeps program source and executable code up-to-date on multiple machines. Unfortunately, *rdist* is a new feature of 4.3 BSD Unix, and hard-copy documentation does not exist, but documentation is available on-line to 4.3 BSD Unix users. *Cc*, *lint*, and *make* are also explained in [Computer84a].

2.2.3. Post-processing the Data

After combining the trace data from the C Shell and the accounting files, the data needs to be post-processed for easier analysis. Each command is assigned a unique identifier, so this identifier can be used as an index into arrays used in the analysis program. Commands are grouped into categories, and the category for each command is recorded in each record. A flag is set or cleared to indicate whether the command line began execution in the background or the foreground. A unique login session identifier is assigned to each login session. Each user is assigned to a class depending on whether he is a secretary, faculty member, graduate student, guest, programmer, root, miscellaneous, or unknown. The miscellaneous accounts are separated into two classes, a "misc-human" class for users that do not fit any of the other categories, and a "misc-non-human" class for certain programs that have accounts and execute commands from command files. Sponsors for each user are also recorded. Terminal identifiers in the accounting data are converted to classifications, such as remote logins from other machines or dial-ups.

2.2.4. Analysis

The final step is to read the post-processed data and count the number of times the various commands and features are used. Separate counts were maintained for different user classes to determine whether different types of users had different usage patterns. Separate counts were also kept for different command sources, to determine whether usage patterns differed between sources. (Command sources are explained in Section 3.1.2).

Since the data was post-processed, analysis was relatively easy in most cases. The command line was already parsed into separate commands, and command identifiers were already recorded with the commands. So, counting the commands was quite easy. The source for each command line and the user's classification were also already recorded.

Certain data still had to be obtained from the command line, however. For instance, use of pipes and input/output redirection was only available in the command line. This information was easily obtained, though.

3. Implementation Issues

This section explains those issues that surfaced during implementation of my design. While my design included four steps, the implementation required five steps. After the data was gathered from the C Shell in the first step, it had to be sorted before it could be merged with the accounting data. After merging, the data was post-processed and analyzed.

3.1. Tracing the C Shell

3.1.1. Tracing Issues

Very few lines of code were added or modified in the C Shell to allow data to be collected. The major issue in designing the C Shell tracer was deciding what to record. Besides recording the commands, I wanted enough information to identify the user uniquely and correlate the trace data with the accounting data. So, a user identifier and the time were recorded. Since I wanted to study which features of the C Shell were being used, not just which commands, the entire command line was recorded. Also, since I wished to study the use of the history mechanism and alias substitutions, I wanted the command line both before and after history and aliasing had taken place. History substitutions are done before alias substitutions. So, I recorded the command line before history substitutions, after history substitutions but before aliasing, and after aliasing. To minimize the amount of data generated, all three are recorded only if history substitution and aliasing has taken place. Two command lines are recorded if only one of history substitution or aliasing occurred, and one command line is recorded if neither occurred.

I used two techniques to reduce the overhead imposed by the tracing facilities. First, to minimize the number of system calls used to write data to the trace files, data was only written to the trace file after at least 1024 bytes of data accumulated for a given user, or the user logged off. Note that a separate buffer of data is maintained for each user. Second, the number of time() system calls was also minimized. Most users set a C Shell option which notifies them periodically if they have new mail. If the user has specified that his mail file should be checked, every time the C Shell completes a command, just before a new prompt is printed, the C Shell checks to see if it is time to check the user's mail file. To check the time, the C Shell makes a time() system call. So, if the user has set this option, the time returned by this system call can be used in the C Shell trace record. Note that this time is the time after all commands in the command line have been executed, unless the command line was executed in the background. For background command lines, the time will be the time the command line was typed in. Note that this guarantees only that the time field will contain a value between the time the command line began execution and the time the command completed execution.

Write synchronization problems were handled by opening the trace file in append mode which guarantees that each write to the trace file will be appended to the end of the trace file. Also, writes always included whole records, never incomplete records.

3.1.2. Trace Data Formats

The format of the C Shell trace data formats was relatively simple and is detailed in Appendix B. Each record includes an integer user identifier, the time, the source for the command, whether or not the command line contained a syntax error, and the command line itself. The integer user identifier is assigned by the system for internal use. The source for the command is either a command file or the user typing at his terminal. If the command file is a *.login*, *.cshrc*, or *.logout* file, this is also indicated. (The purposes of *.login*, *.cshrc*, and *.logout* files are explained in Appendix A.) One, two, or three versions of the command line are recorded, as explained above.

3.2. Sorting

Since the C Shell data was only actually written to the trace file after at least 1024 bytes had accumulated, and separate buffers of data were maintained for each user, the data in the trace file was not in any strict order. So, the trace file records had to be sorted by time before they could be merged with the accounting file records.

3.3. Combining C Shell Data with Accounting Data

Combining the C Shell data with the accounting data was, by far, the most complex step to implement.

3.3.1. Matching Records

C Shell records can be matched to accounting records by using the times, the user identifier, and the command name. Recall that the entire command line was recorded from the C Shell, and each command line may contain multiple commands. So, the commands must be extracted from the command line, and then matched to the corresponding accounting records. While in most cases the matching is quite straight-forward, certain records introduce problems.

3.3.1.1. Unmatchable C Shell Records

Some C Shell records cannot be matched to any accounting data. There are four possible reasons for this. The first and most common reason is that the C Shell record may contain only built-in commands, which do not have corresponding accounting records. Built-in commands are executed directly by the C Shell, without calling another program. A second reason is that the record may contain only misspelled commands. In this case, the command(s) was (were) never executed and never generated accounting data. A third possible reason is that the command was the name of a Bourne Shell *(sh)* script, and as explained in Section 2.2.2, these commands are not matched to accounting data. The last possible reason is that accounting was "turned off" when the C Shell data was generated. This happened when 98 per cent of the usable space on the file system that contained the accounting file was in use. Accounting was "turned on" again when no more than 96 per cent of the usable space on the file system was in use.

3.3.1.2. Unmatchable Accounting Records

Some accounting records do not correspond to any C Shell records. There are three possible reasons why this may happen. First, if the command corresponds to a Bourne Shell *(sh)* script, it is not matched to C Shell data, as explained in Section 2.2.2. Second, the command may have been executed from another shell that was not traced, such as the Bourne shell. Third, in Unix, any process may execute an exec() system call [Computer84a] to create another process. This creation and execution of another process will generate another accounting record, but, if it does not correspond to anything the user typed in the C Shell, it will not be recorded in the C Shell trace data.

3.3.1.3. Multiple-Accounting Records

As explained in Section 2.2.2, certain C Shell commands generate multiple accounting commands. In this case, it is impossible to determine which accounting commands correspond to a given C Shell command. Instead the matching program had to guess which accounting commands correspond to the given C Shell command. For example, assume the C Shell trace file contains a *make* command for user `x`. First, all of user `x`'s C Shell commands that match one-to-one to accounting commands are matched together. Then the remaining accounting commands that were executed by user `x` in the same time frame that the *make* command was executed are matched to the *make* C Shell command.

In most cases, my algorithm for matching accounting commands to these commands that generate multiple accounting commands works quite well. In a few cases, however, the algorithm does not work well at all. If the user types more than one command in the same time frame that generates multiple-accounting records, the matching program is unable to determine which C Shell command the accounting records should be matched to. My algorithm will use the time on each record to match the records, but as the times on the C Shell records get closer together, it becomes more difficult for the algorithm to determine which C Shell command to match the accounting records to. This problem becomes especially severe when a user runs two or more C Shell commands at the same time that generate multiple-accounting records. Fortunately, this case occurs relatively infrequently.

The user *root* creates a second problem when attempting to match commands that generate multiple-accounting records, as many accounting commands will be incorrectly matched to the multiple-accounting records. The algorithm handles unmatched accounting commands specially: for any user, it looks for accounting commands that do not match C Shell data but were executed near the time that a multiple-accounting record was generated; these accounting commands are matched to the multiple-accounting record. This works quite well for ordinary users, but it does not work well for root. Root runs many processes, such as *sendmail*, that are not matched to any C Shell data. My algorithm will match these accounting commands to the multiple-accounting records, while many of them should not be matched. I was unable to solve this problem, but, obviously, this problem only affects *root*.

3.3.1.4. Matching Times

Trying to match the times on the C Shell and accounting records was difficult. The main problem was that corresponding records did not necessarily appear in the same order in the accounting data file and the trace data file. The accounting file is sorted by the time the command completed execution, not the time the command began execution. The problem with the C Shell time, however, is that the time is neither the starting time nor the ending time for the command line, but something in between, as noted above in Section 3.1.1.

The format of the accounting file made sorting difficult because it is a binary format, not an ASCII format. As a result, the standard system sort program, which requires ASCII records, could not be used to sort the accounting file by start time.[1] So, if the accounting file was going to be sorted, a sort program would have to be written. Since the C Shell data file is an ASCII file, the standard system sort program could be used to sort it. As a result, I decided to sort the C Shell data and avoid sorting the accounting file, if possible.

My solution to these problems is somewhat confusing. The accounting file contains both the beginning time and ending time for each command, (Actually, it contains the beginning time and the elapsed time, so the ending time can be deduced.) while the tracefile contains only one time that is guaranteed to be somewhere between the beginning time and the ending time. Consequently, if the records were processed in increasing time order, an accounting record would be read in at beginning time, and kept in main memory until it was matched to a trace file record, or the accounting record ending time was reached. Since the trace record time is guaranteed to be between the accounting beginning time and ending time, the accounting record would be in main memory when the corresponding trace record was read. If the accounting record ending time was reached without finding a trace record to match, then a matching trace record did not exist.

---

[1]The standard system sort program expects ASCII records separated by end-of-line characters. Consequently, whenever a byte of an integer field happens to match the bit pattern for an end-of-line character, the sort program will produce invalid results.

The accounting records, however, are not ordered by beginning time, but by ending time, and I wanted to avoid sorting the accounting file, if possible. Consider what happens if the records are processed in *decreasing* time order. Each accounting record would be read in at accounting ending time and kept in main memory until matched to a trace file record, or the accounting beginning time was reached. Again, since the trace record time is guaranteed to be between the accounting beginning time and ending time, the accounting record would be in main memory when the corresponding trace record was read. If the accounting record beginning time was reached without finding a trace record to match, then a matching trace record did not exist.

So, the tracefile is sorted in REVERSE order, and then the accounting file is read in REVERSE order during processing. The accounting file can easily be read in reverse order, by using seeks, as all accounting records are the same size. (Actually, 512 bytes were read at one time, to avoid the overhead of many seek and read operations.)

3.3.2. Combined Data Format

Appendix D lists the format of the data records after the C Shell data and the accounting data have been combined, but before the data has been post-processed. Appendix C lists the format of the accounting records. Those data items not copied directly from the C Shell data records or the accounting records, but added during the combine stage are indicated by asterisks. Many items are self-explanatory, but those that are not are explained below.

3.3.2.1. Record Type

The first byte indicates whether the record contains data from the accounting file and the C Shell, or just one or the other. An "Accounting File Record" contains data from the accounting file, but does not contain data from the C Shell. The reverse is true of "C Shell Only" records. They contain data from the C Shell, but do not contain data from the accounting file. Combined records contain data from the C Shell and the accounting file. A fourth record type is used for keeping track of login sessions. These records are called C Shell accounting records and are explained in Section 3.4.3.

3.3.2.2. Sub-Records

The format of the records that contain C Shell data is a complex tree structure. Each command line corresponds to one record. Recall that the command line from the C Shell may contain more than one command. For example, the command line "make; echo ^G" contains two commands, *make* and *echo*. This command line will compile source files as necessary (*make*) and then (;) signal with a bell (^G) when finished. Each command corresponds to a sub-record or branch within the record for the command line. So, *make* and *echo* will each correspond to a branch in the tree structure.

Command sub-records are classified in four ways. Each command is either an accounting command, a multiple-accounting command, a built-in command, or a missing command. Accounting and multiple-accounting commands are those commands in C Shell command lines that have corresponding accounting records. Built-in commands are executed directly by the C Shell and do not have corresponding accounting records. Missing commands are commands in C Shell command lines that do not have corresponding accounting records, nor are they built-in commands. The causes for missing commands were explained in Section 2.2.2. In the above example, *make* is a multiple-accounting command, as it may generate many accounting records while it compiles files, and *echo* is a built-in command, as it is executed directly by the C Shell.

Accounting sub-records and multiple-accounting sub-records contain the corresponding record(s) from the accounting file. C Shell commands that correspond to multiple

accounting records are called multiple-accounting commands. Recall that these commands were explained in Section 2.2.2. All other C Shell commands that correspond one-to-one to accounting records are just called accounting commands. Note that the accounting count field contains the total number of accounting and multiple-accounting sub-records. A multiple-accounting sub-record that contains several accounting records is counted only once in this accounting count field. In the above example, the accounting count field will contain a one, since the command line contains one multiple-accounting sub-record, the *make* command. Multiple-accounting sub-records are themselves broken into sub-sub-records. Each multiple-accounting sub-record contains one command from the C Shell and one or more (usually several) accounting records. In the above example, the *make* sub-record will contain all the accounting records resulting from compiling the necessary source files.

3.4. Post-processing

The post-processing step adds additional redundancy to the data to make the data easier to analyze. This step was relatively straight-forward. The format for the data records after post-processing is given in Appendix E. Most data items are simply copied during post-processing. Those data items that are added during post-processing are noted with asterisks and explained below.

3.4.1. New User Identifiers

New user integer identifiers were added during post-processing. The problem with the system integer identifiers is that they are not continuous, nor are they the same on each machine for the same user. I wished to use these identifiers as indices into arrays during analysis, so I wanted continuous integers, to avoid large, sparse arrays. In addition, I wished to allow the data gathered to be merged with data gathered later on other machines. Many users have accounts on more than one machine, but they are not always assigned the same unique user identifier on each machine. I wished to assign each user the same unique user identifier, regardless of which machine he was using. If machine-unique user identifiers are ever needed, they can be created by catenating the machine identifier with the user identifier.

3.4.2. User Classifications

Users were classified by the type of work they were doing and who was sponsoring their account. If the user is a staff member, the sponsor is simply staff. Otherwise, the sponsor is the professor that is sponsoring the account. The nine user classifications are Secretary, Faculty, Graduate Student, Guest, Misc-Human, Misc-Non-Human, Programmer, Root, and Unknown. Guest accounts are commonly given to people requesting temporary or occasional access to a machine. Those people who did not seem to fit into any particular category were assigned to the Misc-Human class. A few programs, such as *who* program, have accounts. These programs were assigned to the Misc-Non-Human class. Root was given a separate class, as root's usage patterns were expected to be unlike any other class of users. Since the classifications were done by hand, a few users were assigned to the Unknown classification, as I was unable to determine their correct classification. A very small number of undergraduates involved in research projects also have accounts. These undergraduates were assigned to the graduate student category, as their usage patterns were expected to be quite similar to graduate students.

3.4.3. Login Session Identifiers

A unique login session identifier was assigned to each login session. To assign these login session identifiers, I used the *csh* records in the accounting file, each of which corresponds to an execution of the C Shell. Since each login session corresponds to an

execution of the C Shell, each login session should have a *csh* record in the accounting file. Unfortunately, login sessions do not always have *csh* records in the accounting file, and *csh* records correspond to other events besides login sessions. For example, each C Shell script corresponds to a separate execution of the C Shell. Users may also execute commands remotely from another machine without actually logging in, creating a separate execution of the C Shell. To handle the multiple *csh* records in the accounting file, whenever I encounter multiple *csh* records for the same user, I use only the record with the longest elapsed time, as this record should correspond to a login session.

Note that this means that I do not distinguish between concurrent login sessions for the same user. This is really an advantage, since some users have SUN workstations, and log onto the machine several times from different windows, creating multiple concurrent login sessions. This also implies that if a user is logged onto multiple terminals at the same time, all login sessions will be recorded as one login session.

Csh records will be missing from the accounting file if accounting was turned off as explained in Section 2.3.1.1 or if the machine crashed while the user was logged on. To handle login sessions that do not have *csh* records in the accounting file, I create login sessions. Whenever a C Shell record is read for a user that does not have a current login session, I create a login session. To distinguish between these "unknown" login sessions and "known" login sessions, I assign negative integer identifiers to "unknown" login sessions and positive integer identifiers to "known" login sessions. So that these identifiers may still be used as indices into arrays during analysis, I ensure that the absolute value of the identifiers is unique.

Another problem with creating the login session identifier results from my algorithm to combine the accounting and C Shell trace files. If processed in the usual manner that all accounting records are processed during the combine phase, the *csh* accounting record for a login session will not appear in the combined data file until *after* all the data for that login session has been processed. (Note that the login session identifier is created during the post-processing stage.) During the combine stage, since the *csh* accounting record for a login session does not correspond to anything the user typed, it will not be matched to a trace file record. It will be read in at accounting ending time, held in main memory until accounting beginning time, and then printed in the combined data file. So, during the post-processing stage, the *csh* accounting record for a login session will appear in the file of combined data AFTER all commands for that login session. I solved the problem by writing each *csh* record twice in the combine stage, once as an "accounting" record, and once an a "csh" type record. The "accounting" type record will be printed when matched to a C Shell trace record or the execution beginning time is reached, just like other accounting records are processed. The "csh" type record, however, will be printed *as soon as the csh accounting record is read*, so it will appear in the combined data file BEFORE the commands for the corresponding login session. This *csh* type record can then be used to create the login session during post-processing, as it will be read before any other data for the login session is read.

3.4.4. Command Identifiers and Categories

Each command was assigned a unique integer identifier and a category. These identifiers are used as indices into arrays during analysis. Each command that is not a built-in command resides in a particular directory. These directories were used to create the command categories. Commonly-used directories were each given a separate category, and built-in commands were assigned to another category. Non-built-in commands that were not found in one of the commonly-used directories were assigned to the "other" category. A user-created program is an example of a command that would not be found in one of the commonly-used directories.

3.4.5. Terminal Classifications

Terminals were classified by whether they were hardwired to the machine (this is the typical case), attached to a patchboard (a mechanism for connecting to various machines), remote logins from another machine, dialups, the system console, or other. Commands that are not associated with a terminal, such as system programs that run automatically, are given the "other" terminal classification. All classifications could be determined by examining the *tty* field in the accounting record.

3.4.6. Background/Foreground Flag

A flag was added during post-processing to indicate whether the command line began execution in the background or the foreground. This can easily be determined by examining the last character in the command line. If this character is a "&," then the command line began execution in the background. Note that I only record whether the command line BEGAN execution in the background or the foreground. If the user later moved the command(s) from the background to the foreground, or vice versa, I did not record this information.

3.5. Analysis Implementation

The analysis was one of the easier steps to implement. This simply involved reading the data and incrementing counters. Note that I still need to read the command line to get pipes, input/output redirection, "||," and "&&." While this information could have been read during post-processing and fields added to the post-processed records to contain this information, this seemed like a needless repeat of information. Even if I had added these fields to the post-processed records, I would still have needed to read the command line to determine where the pipes and input/output redirection were. I would also have needed to read the command line to get command parameters, or any other information I might later decide to analyze.

4. Trace Analysis

This section explains my results. First my general results are given. Then more detailed results for specific commands are given. Finally, my results are compared to previous studies.

4.1. General Results

Usage patterns were relatively simple. Almost all command lines contain only one command. The average user invokes only about one-fifth of all commands. Alias and history substitutions are the most commonly used features of the C Shell, with 28.5% of command lines typed at terminals using aliasing, and 5.0% of command lines typed at terminals using history substitutions. Both alias and history substitutions decrease the number of user keystrokes.

Many results are quite predictable. Commands that use a large amount of CPU time are often executed in the background. Complex features, such as "||" and "&&," are very seldom used.

One result that was not predicted was that 28.4% of the data was generated by *daemon*, a special "user" that executes system command files.

4.1.1. Distribution of Data

The distribution of the data by the users' work type, or classification, is give in Table 1, "Number of Commands from Each Class." The "Count" column gives the number of commands for that user classification, and the "Per Cent of Total" column gives the ratio of the numbers of commands for that user classification to the total number of commands.

Note that almost 30 per cent of the data is from the "Misc-Non-Human" category.

Table 1 - Number of Commands from Each Class		
Class	Count	Per Cent of Total
Secretary	31711	6.6%
Facu	62317	13.0%
Grad	99220	20.7%
Guest	42583	8.9%
Misc-Human	36623	7.7%
Misc-Non-Human	140294	29.3%
Prog	54117	11.3%
Root	11671	2.4%
Unknown	—	—
Total	478536	100.0%

Table 2, "Misc-Non-Human Command Details," shows the miscellaneous-non-human users and the number of commands executed by each. Note that *daemon* executes 96.9% of all Misc-Non-Human commands and 28.4% of all users' commands. While I was unable to determine exactly what *daemon* was doing, it appeared that *daemon* was performing functions that could also have been done in a program without using the C Shell, such as processing command line arguments for local text formatting programs. Consequently, this large percentage of data generated by *daemon* appears to be a local anomaly.

Table 2 - Misc-Non-Human Command Details			
Login Id	Command Count	Per Cent of Misc-Non-Human Commands	Per Cent of Total Commands
daemon	135912	96.9%	28.4%
uucp	4330	3.1%	0.9%
nobody	52	0.0%	0.0%
Totals	140294	100.0%	29.3%

4.1.2. User Vocabulary

Most users commonly use only a small percentage of available commands, or, in other words, they have a relatively small *vocabulary*. Table 3, "Vocabulary," shows the average number of commands used by each user in each class during the two week period that the data was collected. Commands that were used by less than five users were omitted from the calculations for this table, as these commands were likely to be obscure or not available to the public. The average user uses only about 20 per cent of the available commands. Note that *root* has the most diverse vocabulary. As might be expected, staff programmers have a relatively diverse vocabulary, and guests do not have a very diverse vocabulary. "Misc-Non-Humans" execute only a few commands, as each is really just a program intended to perform a specific task(s).

Table 3 - Vocabulary			
Average Number of Commands Used*			
Classes	No. Users	Av. No. Commands Used Per User	Per Cent of Common Commands
Secretary	15	51	20.5%
Facu	14	48	19.3%
Grad	62	48	19.3%
Guest	37	43	17.3%
Misc-Human	13	59	23.7%
Misc-Non-Human	3	12	4.8%
Prog	21	56	22.5%
Root	1	147	59.0%
Unknown	—	—	—
Overall	166	49	19.7%

*Commands used by less than five users are not included.

4.1.3. Command Sources

C Shell data was classified according to the source of the commands: *terminal*, *.login*, *.cshrc*, *.logout*, and *command files*. *Terminal* is simply input from the user typing at his terminal. The *.login*, *.cshrc*, and *.logout* files are special types of command files, and they are commands that are not typed by the user at his terminal. For an explanation of command files, consult Appendix A. The distribution of the number of commands from each source is given in Table 4, "Number of Commands from Each Source." In each box, the first entry indicates the total number of commands for that classification and source. The second entry is the ratio of the first entry to the total number of commands, and the third entry is the ratio of the first entry to the total number of commands for that classification. For example, the upper left box indicates that secretaries typing at their terminals executed 9,821 commands. These commands represent 2.1% of all commands and 31.0% of all secretaries' commands. Note that almost half of all commands come from command files, but most command file commands are from "Misc-Non-Humans." Note also that there are more commands from the *.login* files and the *.cshrc* files than there are from the terminal. Since most or all commands in a user's *.login* file and *.cshrc* file are to set up his environment, this implies that users are willing to invest time when they log onto the machine to change their environment to suit their preferences. A second table, Table 29 showing the same information for command lines instead of commands is given in Appendix F. The information in Table 29 is quite similar to Table 4, but information from both tables is used to derive different statistics in tables to follow.

Table 4 - Number of Commands from Each Source						
	Terminal	.login	.cshrc	.logout	Command File	Total
	9821	4254	10259	571	6806	31711
	2.1%	0.9%	2.1%	0.1%	1.4%	6.6%
Secretary	31.0%	13.4%	32.4%	1.8%	21.5%	100.0%
	14227	5010	15736	449	26895	62317
	3.0%	1.0%	3.3%	0.1%	5.6%	13.0%
Facu	22.8%	8.0%	25.3%	0.7%	43.2%	100.0%
	26440	22585	37865	1716	10614	99220
	5.5%	4.7%	7.9%	0.4%	2.2%	20.7%
Grad	26.6%	22.8%	38.2%	1.7%	10.7%	100.0%
	14597	10068	14719	695	2504	42583
	3.1%	2.1%	3.1%	0.1%	0.5%	8.9%
Guest	34.3%	23.6%	34.6%	1.6%	5.9%	100.0%
	11236	6020	12306	149	6912	36623
	2.3%	1.3%	2.6%	0.0%	1.4%	7.7%
Misc-Human	30.7%	16.4%	33.6%	0.4%	18.9%	100.0%
	1	—	2	—	140291	140294
	0.0%	—	0.0%	—	29.3%	29.3%
Misc-Non-Human	0.0%	—	0.0%	—	100.0%	100.0%
	11865	12859	15378	813	13202	54117
	2.5%	2.7%	3.2%	0.2%	2.8%	11.3%
Prog	21.9%	23.8%	28.4%	1.5%	24.4%	100.0%
	2587	—	4731	—	4353	11671
	0.5%	—	1.0%	—	0.9%	2.4%
Root	22.2%	—	40.5%	—	37.3%	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	90774	60796	110996	4393	211577	478536
	19.0%	12.7%	23.2%	0.9%	44.2%	100.0%
Totals	19.0%	12.7%	23.2%	0.9%	44.2%	100.0%

4.1.4. Simplicity of Usage Patterns

Most uses of the Unix C Shell are relatively simple. The vast majority of all command lines contain only one command, as shown in Table 5, "Number of Commands in Each Command Line." More detailed statistics regarding command lines with multiple commands are given in Section 4.2.2.

Table 5 - Number of Commands in Each Command Line						
No. of Commands	1	2	3	4	5 or more	Total No. Command Lines
No. Comm Lines	435522	10993	5875	450	424	453264
Per Cent of Total	96.1%	4.9%	3.9%	0.4%	0.1%	100.0%

4.1.5. Command Line Expansions

Users often use *aliases* and *history substitutions* to decrease the amount of typing necessary. *Aliases* are a mechanism for substituting a short character string for a longer character string. *Aliases* are often declared in *.cshrc* and *.login* files, as shown in Table 6, "Declaration of Aliases." In each box, the first entry indicates the total number of alias declarations for that classification and source. The second entry is the ratio of the first entry to the total number of commands for that classification and source, and the third entry is the ratio of the first entry to the total number of aliases declared for that classification. For example, the "Guest/.cshrc" box indicates that guests executed 10,271 commands in their .cshrc files, which represents 69.8% of all commands in their .cshrc files and 90.0% of all aliases declared by guests. Note that most of the commands in users' *.cshrc* files are declarations of aliases, and, as indicated in the totals box, alias declarations account for almost one-fourth of all commands. Few aliases are declared by the user typing at his terminal.

Table 6 - Declaration of Aliases

	Terminal	.login	.cshrc	.logout	Command File	Total
	26	259	8615	—	296	9196
	0.3%	6.1%	84.0%	—	4.3%	29.0%
Secretary	0.3%	2.8%	93.7%	—	3.2%	100.0%
	49	495	12980	—	188	13712
	0.3%	9.9%	82.5%	—	0.7%	22.0%
Facu	0.4%	3.6%	94.7%	—	1.4%	100.0%
	30	7237	28892	—	376	36535
	0.1%	32.0%	76.3%	—	3.5%	36.8%
Grad	0.1%	19.8%	79.1%	—	1.0%	100.0%
	18	883	10271	—	234	11406
	0.1%	8.8%	69.8%	—	9.3%	26.8%
Guest	0.2%	7.7%	90.0%	—	2.1%	100.0%
	45	2350	9993	—	2649	15037
	0.4%	39.0%	81.2%	—	38.3%	41.1%
Misc-Human	0.3%	15.6%	66.5%	—	17.6%	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Misc-Non-Human	—	—	—	—	—	—
	38	4591	11604	—	519	16752
	0.3%	35.7%	75.5%	—	3.9%	31.0%
Prog	0.2%	27.4%	69.3%	—	3.1%	100.0%
	11	—	4156	—	—	4167
	0.4%	—	87.8%	—	—	35.7%
Root	0.3%	—	99.7%	—	—	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	217	15815	86511	—	4262	106805
	0.2%	26.0%	77.9%	—	2.0%	22.3%
Totals	0.2%	14.8%	81.0%	—	4.0%	100.0%

Aliases are also used quite frequently, especially by users typing at their terminals. Table 7, "Use of Aliases," indicates that over 13 per cent of all command lines use aliases.

The table is organized in the same way as Table 6; for example, the upper left box in Table 7 indicates that 25.6% of all commands typed by secretaries at their terminals used aliases, and 80.3% of all aliases used by secretaries were typed at the terminal. Note that, of users typing at their terminal, guests have a much lower-than-average use of aliases. "Root" has the highest use of aliases, at the terminal, which is logical, since "root" is actually those users currently using their super-user privileges. Super-user privileges are not commonly given to or used by the less-sophisticated user.

Table 7 - Use of Aliases

	Terminal	.login	.cshrc	.logout	Command File	Total
	2387	60	507	20	—	2974
	25.6%	1.6%	5.0%	3.5%	—	9.7%
Secretary	80.3%	2.0%	17.0%	0.7%	—	100.0%
	4709	113	7937	12	175	12946
	36.3%	3.0%	50.5%	2.7%	0.7%	21.8%
Facu	36.4%	0.9%	61.3%	0.1%	1.4%	100.0%
	7822	1960	10011	181	295	20269
	31.9%	9.9%	26.6%	11.0%	2.9%	21.6%
Grad	38.6%	9.7%	49.4%	0.9%	1.5%	100.0%
	2134	305	3235	12	32	5718
	15.3%	3.5%	22.3%	1.7%	1.4%	14.3%
Guest	37.3%	5.3%	56.6%	0.2%	0.6%	100.0%
	2817	1752	4258	—	1937	10764
	26.0%	31.2%	34.7%	—	29.1%	30.3%
Misc-Human	26.2%	16.3%	39.6%	—	18.0%	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Misc-Non-Human	—	—	—	—	—	—
	3454	54	3322	6	100	6936
	30.8%	0.5%	21.7%	0.8%	0.8%	13.3%
Prog	49.8%	0.8%	47.9%	0.1%	1.4%	100.0%
	926	—	—	—	5	931
	40.9%	—	—	—	0.1%	8.2%
Root	99.5%	—	—	—	0.5%	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	24249	4244	29270	231	2544	60538
	28.5%	8.0%	26.5%	5.4%	1.3%	13.4%
Totals	40.1%	7.0%	48.3%	0.4%	4.2%	100.0%

Although not used as often as aliases, *history substitutions* are also used. History substitutions allow a user to repeat a previous command line, or a command line similar to a previous command line, with only a few keystrokes. They are used almost exclusively by users typing at their terminals. Statistics regarding the use of history substitutions are given in Table 8, "Use of History." It is not surprising that "root" and faculty members have a higher-than-average use of history substitutions and secretaries have a lower-than-average use of history substitutions. It is surprising, however, that graduate students and programmers have a lower-than-average use of history substitutions.

Table 8 - Use of History			
Class	Frequency	Per Cent of all Terminal Lines for Class	Per Cent of All Command Lines for Class
Secretary	375	4.0%	1.2%
Facu	791	6.1%	1.3%
Grad	990	4.0%	1.1%
Guest	693	5.0%	1.7%
Misc-Human	919	8.5%	2.6%
Misc-Non-Human	—	—	—
Prog	356	3.2%	0.7%
Root	163	7.2%	1.4%
Unknown	—	—	—
Totals	4287	5.0%	0.9%

Using aliasing and history substitutions decreases the number of characters that the user would otherwise have needed to type and remember. Table 9, "Before/After Expansion Ratio," shows the ratio of the length of the command line before alias and history substitutions to the length of the command line after alias and history substitutions, regardless of whether those command lines used alias or history substitutions. Note that the ratio is lowest, indicating the greatest expansion, for users typing at their terminals. As might be expected, users are more concerned about conserving key strokes at their terminal, where command lines are both typed and used once, than they are in command files, where command lines are only typed once, but used many times. Before/after expansion ratios for command lines using history substitutions are given in Table 10, "Before/After History Ratio," and before/after expansion ratios for command lines using alias substitutions are given in Table 11, "Before/After Alias Ratio." Note that the lowest ratio, indicating the greatest expansion, is for users the use history substitutions typing at their terminals. Tables showing the length of the user's command line are given in Appendix G.

Table 9 - Before/After Expansion Ratio						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	0.81	0.99	0.99	0.98	1.00	0.97
Facu	0.72	0.97	0.92	0.99	1.00	0.94
Grad	0.73	0.97	0.96	0.95	0.99	0.94
Guest	0.76	0.98	0.97	0.99	1.00	0.95
Misc-Human	0.77	0.97	0.95	1.00	0.95	0.93
Misc-Non-Human	1.00	—	1.00	—	1.00	1.00
Prog	0.73	0.99	0.96	1.00	1.00	0.96
Root	0.69	—	1.00	—	1.00	0.96
Unknown	—	—	—	—	—	—
Overall	0.75	0.98	0.96	0.98	1.00	0.96

Table 10 - Before/After History Ratio						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	0.34	—	—	0.88	—	0.37
Facu	0.31	—	—	—	—	0.31
Grad	0.30	—	—	—	—	0.30
Guest	0.35	—	—	—	—	0.35
Misc-Human	0.30	—	—	—	—	0.30
Misc-Non-Human	—	—	—	—	—	—
Prog	0.33	—	—	—	—	0.33
Root	0.35	—	—	—	—	0.35
Unknown	—	—	—	—	—	—
Overall	0.32	—	—	0.88	—	0.32

Table 11 - Before/After Alias Ratio						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	0.51	0.59	0.84	0.83	—	0.61
Facu	0.48	0.21	0.85	0.33	0.72	0.75
Grad	0.40	0.73	0.85	0.49	0.80	0.72
Guest	0.26	0.61	0.88	0.39	0.79	0.72
Misc-Human	0.47	0.83	0.83	—	0.85	0.78
Misc-Non-Human	—	—	—	—	—	—
Prog	0.39	0.41	0.82	0.50.	0.83	0.67
Root	0.47	—	—	—	0.50	0.47
Unknown	—	—	—	—	—	—
Overall	0.42	0.71	0.85	0.52	0.83	0.73

4.1.6. Accounting Statistics

The average use of system resources for all commands is given in Table 12, "Overall Accounting Statistics." Note that the average command uses a very small amount of CPU time. The average elapsed time is highly skewed by a few commonly used interactive commands that have high idle times, such as editors. Table 13, "Accounting Statistics for Most Commands," shows the same data, but omits nine commonly-used commands with high idle times.

Table 12 - Overall Accounting Statistics	
	Average
System time (minutes:seconds.tenths)	0:01.1
User time (minutes:seconds.tenths)	0:02.0
Elapsed time (minutes:seconds.tenths)	4:55.5
Memory Usage (512 byte units)	55
I/O Operations	23

Table 13 - Accounting Statistics for Most Commands*	
	Average
System time (minutes:seconds.tenths)	0:01.0
User time (minutes:seconds.tenths)	0:02.5
Elapsed time (minutes:seconds.tenths)	1:25.3
Memory Usage (512 byte units)	50
I/O Operations	18

*Nine common commands with high idle times (csh, mail, man, more, msgs, rlogin, rsh, sh, and vi) have been omitted.

4.2. Detailed Results

This section gives more detailed results than those presented in the previous section.

4.2.1. Commands Per Login Session, Command File, and Day

The average number of commands in command files is given in Table 14, "Average Number of Commands in File." Note that, by far, the largest number of commands per command file is associated with "Misc-Non-Humans." This number probably does not reflect reality. My analysis program detects the end of a command file when it processes a non-command-file command, but Misc-Non-Humans execute command files almost exclusively. As a result, my analysis program may process commands from many command files, but assume that the commands are all from the same command file.

Table 14 - Average Number of Commands in File				
	.login	.cshrc	.logout	Command File
Secretary	12	28	2	20
Facu	11	25	2	29
Grad	13	22	2	12
Guest	10	16	2	15
Misc-Human	10	20	1	41
Misc-Non-Human	—	1	—	23381
Prog	14	21	1	36
Root	—	25	—	23
Unknown	—	—	—	—
Overall	12	21	2	72

The average number of commands in each login session is given in Table 15, "Average Number of Commands in Login Session." The average number of commands in a "misc-non-human" login session is zero, as only commands typed by users at terminals are included in these figures, and the vast majority of "misc-non-human" commands come from command files.

Table 15 Average Number of Commands in Login Session	
Secretary	43
Facu	68
Grad	29
Guest	25
Misc-Human	31
Misc-Non-Human	0
Prog	27
Root	69
Unknown	—
Overall	21

Table 16, "Average Number of Terminal Commands Per User Per Day," shows the average number of commands typed per user per day. Note that this number varies widely between classifications.

Table 16 Average Number of Terminal Commands Per User Per Day	
Secretary	104
Facu	161
Grad	74
Guest	71
Misc-Human	100
Misc-Non-Human	1
Prog	73
Root	184
Unknown	—
Overall	88

4.2.2. More Details on Simple Usage Patterns

As explained in Section 4.1.4, users have relatively simple usage patterns. Table 17, "Use of Sequential Execution," shows that sequential execution is used in less than 3 per cent of all command lines. Sequential execution allows a user to enter two or more commands on the same command line to be executed sequentially. In each box, the first entry indicates the number of command lines using sequential execution for that classification and source. The second entry indicates the ratio of the first entry to all command lines for that classification and source, and the third entry indicates the ratio of the first entry to the total number of command lines using sequential execution for that classification. For example, the upper-left box indicates that secretaries typing at their terminals used sequential execution in 95 command lines, which represents 1.0% of all command lines typed by secretaries at their terminals. Also, 16.5% of all secretaries' command lines that used sequential execution were typed at their terminals. Note that while overall use of sequential execution is relatively rare, sequential execution is used frequently in *.login* files. Approximately one-third of faculty members' *.login* file commands

use sequential execution, and over 10 per cent of all users' *.login* file commands use sequential execution.

Table 17 - Use of Sequential Execution (';')

	Terminal	.login	.cshrc	.logout	Command File	Total
	95	444	6	21	10	576
	1.0%	11.7%	0.1%	3.7%	0.2%	1.9%
Secretary	16.5%	77.1%	1.0%	3.6%	1.7%	100.0%
	549	1315	20	—	56	1940
	4.2%	34.5%	0.1%	—	0.2%	3.3%
Facu	28.3%	67.8%	1.0%	—	2.9%	100.0%
	1140	2536	10	55	90	3831
	4.6%	12.8%	0.0%	3.4%	0.9%	4.1%
Grad	29.8%	66.2%	0.3%	1.4%	2.3%	100.0%
	333	1041	9	—	36	1419
	2.4%	12.1%	0.1%	—	1.6%	3.5%
Guest	23.5%	73.4%	0.6%	—	2.5%	100.0%
	177	365	—	—	184	726
	1.6%	6.5%	—	—	2.8%	2.0%
Misc-Human	24.4%	50.3%	—	—	25.3%	100.0%
	—	—	—	—	2816	2816
	—	—	—	—	2.2%	2.2%
Misc-Non-Human	—	—	—	—	100.0%	100.0%
	359	1036	93	—	75	1563
	3.2%	8.9%	0.6%	—	0.6%	3.0%
Prog	23.0%	66.3%	6.0%	—	4.8%	100.0%
	285	—	—	—	5	290
	12.6%	—	—	—	0.1%	2.6%
Root	98.3%	—	—	—	1.7%	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	2938	6737	138	76	3272	13161
	3.5%	12.7%	0.1%	1.8%	1.6%	2.9%
Totals	22.3%	51.2%	1.0%	0.6%	24.9%	100.0%

Pipes are also used relatively infrequently in command lines. The frequency of pipe usage is detailed in Table 18, "Pipe Usage." In each box, the first entry indicates the number of command lines using that number of pipes for that source. The second entry indicates the ratio of the first entry to all command lines for that source, and the third entry indicates the ratio of the first entry to the total number of command lines using that number of pipes. For example, the upper-left box indicates that 82,700 command lines typed at terminals did not contain any pipes. This number is 97.2% of all command lines typed at terminals and 18.5% of all command lines that did not contain any pipes. Note that over 98 per cent of all commands lines do not use any pipes, and only 0.1 per cent of all command lines have 2 or more pipes. Users of the Unix C Shell, however, often rave about this feature, as it usually eliminates the need to create temporary files and invoke each command separately.

Table 18 - Pipe Usage						
	Terminal	.login	.cshrc	.logout	Command File	Totals
	82700	53073	110275	4287	196495	446830
	97.2%	99.7%	99.9%	99.8%	98.1%	98.6%
Zero Pipes	18.5%	11.9%	24.7%	1.0%	44.0%	100.0%
	2151	136	57	7	3748	6099
	2.5%	0.3%	0.1%	0.2%	1.9%	1.3%
One Pipe	35.3%	2.2%	0.9%	0.1%	61.5%	100.0%
	255	—	—	—	27	282
	0.3%	—	—	—	0.0%	0.1%
Two Pipes	90.4%	—	—	—	9.6%	100.0%
	5	29	—	—	19	53
Three or	0.0%	0.1%	—	—	0.0%	0.0%
More Pipes	9.4%	54.7%	—	—	35.8%	100.0%
	85111	53238	110332	4294	200289	453264
	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Totals	18.8%	11.7%	24.3%	0.9%	44.2%	100.0%

Input/Output redirection use, shown in Table 19, "I/O Redirection." is more common than pipe usage. Similar to Table 18, in each box, the first entry indicates the number of command lines using that number of input/output redirections for that source. The second entry indicates the ratio of the first entry to all command lines for that source, and the third entry indicates the ratio of the first entry to the total number of command lines using that number of input/output redirections. For example, the upper-left box indicates that 83,372 command lines typed at terminals did not contain any input/output redirection. This number is 98.0% of all command lines typed at terminals and 18.8% of all command lines that did not contain any input/output redirection. Note that only 2.3% of all command lines use input/output redirection.

Table 19 - I/O Redirection						
	Terminal	.login	.cshrc	.logout	Command File	Totals
	83372	53042	110291	4191	192156	443052
No I/O	98.0%	99.6%	100.0%	97.6%	95.9%	97.7%
Redirection	18.8%	12.0%	24.9%	0.9%	43.4%	100.0%
	1597	196	41	103	7574	9511
One I/O	1.9%	0.4%	0.0%	2.4%	3.8%	2.1%
Redirection	16.8%	2.1%	0.4%	1.1%	79.6%	100.0%
	142	—	—	—	559	701
Two or More	0.2%	—	—	—	0.3%	0.2%
I/O Redirections	20.3%	—	—	—	79.7%	100.0%
	85111	53238	110332	4294	200289	453264
	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Totals	18.8%	11.7%	24.3%	0.9%	44.2%	100.0%

Two infrequently used features are "&&" and "||" as shown in Table 20, "Command Lines Using '&&'," and Table 21, "Command Lines Using '||.'" As in previous similar tables, in each box, the first entry indicates the number of command lines using that feature for that classification and source. The second entry indicates the ratio of the first entry to all command lines for that classification and source, and the third entry indicates the ratio of the first entry to the total number of command lines using that feature for that

classification. For example, the "Grad/.login" box in Table 20 indicates that 21 command lines executed in graduate students' .login files used the "&&" feature. This represents 0.1% of all command lines executed from graduate students' .login files, and 38.9% of all graduate students' command lines that used the "&&" feature.

Table 20 - Command Lines Using '&&'

	Terminal	.login	.cshrc	.logout	Command File	Total
	—	—	—	—	—	—
	—	—	—	—	—	—
Secretary	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Facu	—	—	—	—	—	—
	1	21	11	2	19	54
	0.0%	0.1%	0.0%	0.1%	0.2%	0.1%
Grad	1.9%	38.9%	20.4%	3.7%	35.2%	100.0%
	—	3	—	—	—	3
	—	0.0%	—	—	—	0.0%
Guest	—	100.0%	—	—	—	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Misc-Human	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Misc-Non-Human	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Prog	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Root	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	1	24	11	2	19	57
	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Totals	1.8%	42.1%	19.3%	3.5%	33.3%	100.0%

Table 21 - Command Lines Using '||'						
	Terminal	.login	.cshrc	.logout	Command File	Total
	—	10	—	—	—	10
	—	0.3%	—	—	—	0.0%
Secretary	—	100.0%	—	—	—	100.0%
	—	22	—	—	527	549
	—	0.6%	—	—	2.0%	0.9%
Facu	—	4.0%	—	—	96.0%	100.0%
	1	1	—	—	—	2
	0.0%	0.0%	—	—	—	0.0%
Grad	50.0%	50.0%	—	—	—	100.0%
	—	2	—	—	—	2
	—	0.0%	—	—	—	0.0%
Guest	—	100.0%	—	—	—	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Misc-Human	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Misc-Non-Human	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Prog	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Root	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	1	35	—	—	527	563
	0.0%	0.1%	—	—	0.3%	0.1%
Totals	0.2%	6.2%	—	—	93.6%	100.0%

4.2.3. Syntax Errors

Syntax errors are quite rare, as shown in Table 22, "Syntax Errors." The organization of this table is similar to the two previous tables. For example, the upper-left box indicates that secretaries typing at their terminals had 5 command lines with syntax errors. This number represents 0.1% of all secretaries' command lines typed at terminals and 4.0% of all secretaries' command lines with syntax errors. Note that while half of all command lines in "misc-non-human" .cshrc files have syntax errors, this is only one command line. Secretaries' .logout files are the only other case where syntax errors are present in more than 1% of all command lines.

Table 22 - Syntax Errors						
	Terminal	.login	.cshrc	.logout	Command File	Total
	5	—	8	112	—	125
	0.1%	—	0.1%	19.9%	—	0.4%
Secretary	4.0%	—	6.4%	89.6%	—	100.0%
	13	—	—	—	3	16
	0.1%	—	—	—	0.0%	0.0%
Facu	81.3%	—	—	—	18.8%	100.0%
	80	22	—	14	—	116
	0.3%	0.1%	—	0.9%	—	0.1%
Grad	69.0%	19.0%	—	12.1%	—	100.0%
	53	15	4	—	—	72
	0.4%	0.2%	0.0%	—	—	0.2%
Guest	73.6%	20.8%	5.6%	—	—	100.0%
	1	—	—	—	—	1
	0.0%	—	—	—	—	0.0%
Misc-Human	100.0%	—	—	—	—	100.0%
	—	—	1	—	2	3
	—	—	50.0%	—	0.0%	0.0%
Misc-Non-Human	—	—	33.3%	—	66.7%	100.0%
	5	—	—	—	1	6
	0.0%	—	—	—	0.0%	0.0%
Prog	83.3%	—	—	—	16.7%	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Root	—	—	—	—	—	—
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	157	37	13	126	6	339
	0.2%	0.1%	0.0%	2.9%	0.0%	0.1%
Totals	46.3%	10.9%	3.8%	37.2%	1.8%	100.0%

4.2.4. Detailed Command Totals for Each Source and Class

Table 23, "One Hundred Commands from Each Source," and Table 24, "One Hundred Commands from Each Class," present detailed command statistics for the one hundred most frequently executed commands. In Table 23, the first entry in each box is the total number of times that command was executed from that source. The second entry is the ratio of the first entry to the total number of times that command was executed from all sources. The third entry is the ratio of the first entry to all commands from that source, and the fourth entry is the cumulative total of the third entries. For example, the "set/terminal" box shows that the set command was typed at terminals 696 times, which represents 1.1% of all uses of the set command. This number is also 0.8% of all commands typed at terminals. Use of the set and alias commands by users typing at their terminals represents 1.0% of all commands typed at terminals. Note that four commands, *alias*, *set*, *if*, and *shift* account for half of all command executions. The *shift* command, however, is used exclusively in

command files.[2] In fact, of the ten most frequently used commands, seven of them receive over half of their use in command files.

The meanings of the entries in the boxes in Table 24 are similar to the meanings of the entries in the boxes in Table 23. Note, for the ten most frequently used commands, five of them receive over half of their use from "misc-non-humans."

[2]The *shift* command will shift the command line arguments for a command to the left, discarding the leftmost command line argument. *Shift variable* will perform the same function on the specified variable.

Table 23 - One Hundred Commands from Each Source (1)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		217	15815	86511	—	4262	106805
		0.2%	14.8%	81.0%	—	4.0%	100.0%
		0.2%	26.0%	77.9%	—	2.0%	22.3%
alias	1	0.2%	26.0%	77.9%	0.0%	2.0%	22.3%
		696	12504	14198	82	34862	62342
		1.1%	20.1%	22.8%	0.1%	55.9%	100.0%
		0.8%	20.6%	12.8%	1.9%	16.5%	13.0%
set	2	1.0%	46.6%	90.7%	1.9%	18.5%	35.3%
		284	3624	2865	358	40954	48085
		0.6%	7.5%	6.0%	0.7%	85.2%	100.0%
		0.3%	6.0%	2.6%	8.1%	19.4%	10.0%
if	3	1.3%	52.5%	93.3%	10.0%	37.8%	45.4%
		—	—	—	—	24089	24089
		—	—	—	—	100.0%	100.0%
		—	—	—	—	11.4%	5.0%
shift	4	1.3%	52.5%	93.3%	10.0%	49.2%	50.4%
		4	188	25	42	18750	19009
		0.0%	1.0%	0.1%	0.2%	98.6%	100.0%
		0.0%	0.3%	0.0%	1.0%	8.9%	4.0%
switch	5	1.3%	52.9%	93.3%	11.0%	58.1%	54.4%
		2	5	7	—	18101	18115
		0.0%	0.0%	0.0%	—	99.9%	100.0%
		0.0%	0.0%	0.0%	—	8.6%	3.8%
goto	6	1.3%	52.9%	93.3%	11.0%	66.7%	58.2%
		2817	1199	14	689	10163	14882
		18.9%	8.1%	0.1%	4.6%	68.3%	100.0%
		3.1%	2.0%	0.0%	15.7%	4.8%	3.1%
echo	7	4.4%	54.8%	93.4%	26.7%	71.5%	61.3%
		135	10662	2009	6	669	13481
		1.0%	79.1%	14.9%	0.0%	5.0%	100.0%
		0.1%	17.5%	1.8%	0.1%	0.3%	2.8%
setenv	8	4.6%	72.4%	95.2%	26.8%	71.8%	64.1%
		11224	—	—	—	3	11227
		100.0%	—	—	—	0.0%	100.0%
		12.4%	—	—	—	0.0%	2.3%
ls	9	16.9%	72.4%	95.2%	26.8%	71.8%	66.5%
		—	897	1461	68	8438	10864
		—	8.3%	13.4%	0.6%	77.7%	100.0%
		—	1.5%	1.3%	1.5%	4.0%	2.3%
endif	10	16.9%	73.8%	96.5%	28.3%	75.8%	68.7%
		1005	6	4	—	6322	7337
		13.7%	0.1%	0.1%	—	86.2%	100.0%
		1.1%	0.0%	0.0%	—	3.0%	1.5%
exit	11	18.0%	73.9%	96.5%	28.3%	78.7%	70.3%

Table 23 - One Hundred Commands from Each Source (2)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		829	162	1541	—	4326	6858
		12.1%	2.4%	22.5%	—	63.1%	100.0%
		0.9%	0.3%	1.4%	—	2.0%	1.4%
end	12	19.0%	74.1%	97.9%	28.3%	80.8%	71.7%
		1723	142	—	141	3725	5731
		30.1%	2.5%	—	2.5%	65.0%	100.0%
		1.9%	0.2%	—	3.2%	1.8%	1.2%
rm	13	20.9%	74.4%	97.9%	31.6%	82.6%	72.9%
		5158	41	—	71	368	5638
		91.5%	0.7%	—	1.3%	6.5%	100.0%
		5.7%	0.1%	—	1.6%	0.2%	1.2%
cd	14	26.5%	74.4%	97.9%	33.2%	82.7%	74.1%
		6	1189	258	—	3919	5372
		0.1%	22.1%	4.8%	—	73.0%	100.0%
		0.0%	2.0%	0.2%	—	1.9%	1.1%
umask	15	26.5%	76.4%	98.1%	33.2%	84.6%	75.2%
		5280	—	—	—	—	5280
		100.0%	—	—	—	—	100.0%
		5.8%	—	—	—	—	1.1%
vi	16	32.4%	76.4%	98.1%	33.2%	84.6%	76.3%
		1174	227	—	115	3586	5102
		23.0%	4.4%	—	2.3%	70.3%	100.0%
		1.3%	0.4%	—	2.6%	1.7%	1.1%
cat	17	33.7%	76.8%	98.1%	35.8%	86.3%	77.4%
		4017	282	65	7	18	4389
		91.5%	6.4%	1.5%	0.2%	0.4%	100.0%
		4.4%	0.5%	0.1%	0.2%	0.0%	0.9%
more	18	38.1%	77.2%	98.2%	35.9%	86.3%	78.3%
		8	141	3	—	4085	4237
		0.2%	3.3%	0.1%	—	96.4%	100.0%
		0.0%	0.2%	0.0%	—	1.9%	0.9%
LABEL	19	38.1%	77.4%	98.2%	35.9%	88.2%	79.2%
		4005	—	—	4	—	4009
		99.9%	—	—	0.1%	—	100.0%
		4.4%	—	—	0.1%	—	0.8%
jobs	20	42.5%	77.4%	98.2%	36.0%	88.2%	80.0%
		89	2417	178	1	998	3683
		2.4%	65.6%	4.8%	0.0%	27.1%	100.0%
		0.1%	4.0%	0.2%	0.0%	0.5%	0.8%
unset	21	42.6%	81.4%	98.3%	36.1%	88.7%	80.8%
		402	63	—	—	2953	3418
		11.8%	1.8%	—	—	86.4%	100.0%
		0.4%	0.1%	—	—	1.4%	0.7%
while	22	43.0%	81.5%	98.3%	36.1%	90.1%	81.5%

Table 23 - One Hundred Commands from Each Source (3)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		96	2769	124	14	244	3247
		3.0%	85.3%	3.8%	0.4%	7.5%	100.0%
		0.1%	4.6%	0.1%	0.3%	0.1%	0.7%
stty	23	43.2%	86.1%	98.4%	36.4%	90.2%	82.2%
		127	—	61	—	2894	3082
		4.1%	—	2.0%	—	93.9%	100.0%
		0.1%	—	0.1%	—	1.4%	0.6%
@	24	43.3%	86.1%	98.5%	36.4%	91.6%	82.8%
		2888	—	—	—	—	2888
		100.0%	—	—	—	—	100.0%
		3.2%	—	—	—	—	0.6%
fg	25	46.5%	86.1%	98.5%	36.4%	91.6%	83.4%
		1	—	—	—	2812	2813
		0.0%	—	—	—	100.0%	100.0%
		0.0%	—	—	—	1.3%	0.6%
ips	26	46.5%	86.1%	98.5%	36.4%	92.9%	84.0%
		—	74	—	—	2577	2651
		—	2.8%	—	—	97.2%	100.0%
		—	0.1%	—	—	1.2%	0.6%
onintr	27	46.5%	86.2%	98.5%	36.4%	94.1%	84.6%
		47	2024	—	1	290	2362
		2.0%	85.7%	—	0.0%	12.3%	100.0%
		0.1%	3.3%	—	0.0%	0.1%	0.5%
eval	28	46.5%	89.5%	98.5%	36.4%	94.2%	85.1%
		2329	24	—	—	—	2353
		99.0%	1.0%	—	—	—	100.0%
		2.6%	0.0%	—	—	—	0.5%
mail	29	49.1%	89.6%	98.5%	36.4%	94.2%	85.5%
		492	747	1048	9	50	2346
		21.0%	31.8%	44.7%	0.4%	2.1%	100.0%
		0.5%	1.2%	0.9%	0.2%	0.0%	0.5%
source	30	49.6%	90.8%	99.4%	36.6%	94.3%	86.0%
		933	76	11	1242	3	2265
		41.2%	3.4%	0.5%	54.8%	0.1%	100.0%
		1.0%	0.1%	0.0%	28.3%	0.0%	0.5%
clear	31	50.7%	90.9%	99.4%	64.9%	94.3%	86.5%
		784	1356	—	1	39	2180
		36.0%	62.2%	—	0.0%	1.8%	100.0%
		0.9%	2.2%	—	0.0%	0.0%	0.5%
msgs	32	51.5%	93.2%	99.4%	64.9%	94.3%	87.0%
		183	—	—	39	1648	1870
		9.8%	—	—	2.1%	88.1%	100.0%
		0.2%	—	—	0.9%	0.8%	0.4%
sleep	33	51.7%	93.2%	99.4%	65.8%	95.1%	87.4%

Table 23 - One Hundred Commands from Each Source (4)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		—	314	355	3	1165	1837
		—	17.1%	19.3%	0.2%	63.4%	100.0%
		—	0.5%	0.3%	0.1%	0.6%	0.4%
else	34	51.7%	93.7%	99.8%	65.9%	95.6%	87.7%
		1739	—	—	—	2	1741
		99.9%	—	—	—	0.1%	100.0%
		1.9%	—	—	—	0.0%	0.4%
ps	35	53.6%	93.7%	99.8%	65.9%	95.6%	88.1%
		943	16	—	—	735	1694
		55.7%	0.9%	—	—	43.4%	100.0%
		1.0%	0.0%	—	—	0.3%	0.4%
mv	36	54.7%	93.7%	99.8%	65.9%	96.0%	88.5%
		—	—	—	—	1483	1483
		—	—	—	—	100.0%	100.0%
		—	—	—	—	0.7%	0.3%
dit-imagen	37	54.7%	93.7%	99.8%	65.9%	96.7%	88.8%
		1267	—	—	—	125	1392
		91.0%	—	—	—	9.0%	100.0%
		1.4%	—	—	—	0.1%	0.3%
rlogin	38	56.1%	93.7%	99.8%	65.9%	96.7%	89.1%
		1276	75	—	—	4	1355
		94.2%	5.5%	—	—	0.3%	100.0%
		1.4%	0.1%	—	—	0.0%	0.3%
la	39	57.5%	93.8%	99.8%	65.9%	96.7%	89.3%
		368	214	—	125	572	1279
		28.8%	16.7%	—	9.8%	44.7%	100.0%
		0.4%	0.4%	—	2.8%	0.3%	0.3%
date	40	57.9%	94.2%	99.8%	68.7%	97.0%	89.6%
		1222	1	—	—	—	1223
		99.9%	0.1%	—	—	—	100.0%
		1.3%	0.0%	—	—	—	0.3%
logout	41	59.2%	94.2%	99.8%	68.7%	97.0%	89.9%
		1164	8	—	—	—	1172
		99.3%	0.7%	—	—	—	100.0%
		1.3%	0.0%	—	—	—	0.2%
u	42	60.5%	94.2%	99.8%	68.7%	97.0%	90.1%
		824	23	—	—	316	1163
		70.9%	2.0%	—	—	27.2%	100.0%
		0.9%	0.0%	—	—	0.1%	0.2%
ruptime	43	61.4%	94.2%	99.8%	68.7%	97.2%	90.3%
		953	170	—	20	14	1157
		82.4%	14.7%	—	1.7%	1.2%	100.0%
		1.0%	0.3%	—	0.5%	0.0%	0.2%
pwd	44	62.5%	94.5%	99.8%	69.2%	97.2%	90.6%

Table 23 - One Hundred Commands from Each Source (5)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		1148	—	—	—	—	1148
		100.0%	—	—	—	—	100.0%
		1.3%	—	—	—	—	0.2%
finger	45	63.7%	94.5%	99.8%	69.2%	97.2%	90.8%
		51	68	24	960	2	1105
		4.6%	6.2%	2.2%	86.9%	0.2%	100.0%
		0.1%	0.1%	0.0%	21.9%	0.0%	0.2%
fortune	46	63.8%	94.6%	99.8%	91.0%	97.2%	91.1%
		979	21	—	—	9	1009
		97.0%	2.1%	—	—	0.9%	100.0%
		1.1%	0.0%	—	—	0.0%	0.2%
grep	47	64.9%	94.7%	99.8%	91.0%	97.2%	91.3%
		996	—	—	—	—	996
		100.0%	—	—	—	—	100.0%
		1.1%	—	—	—	—	0.2%
4	48	66.0%	94.7%	99.8%	91.0%	97.2%	91.5%
		—	—	—	—	963	963
		—	—	—	—	100.0%	100.0%
		—	—	—	—	0.5%	0.2%
catdvi	49	66.0%	94.7%	99.8%	91.0%	97.6%	91.7%
		934	—	—	—	19	953
		98.0%	—	—	—	2.0%	100.0%
		1.0%	—	—	—	0.0%	0.2%
rsh	50	67.0%	94.7%	99.8%	91.0%	97.6%	91.9%
		2	—	—	—	946	948
		0.2%	—	—	—	99.8%	100.0%
		0.0%	—	—	—	0.4%	0.2%
dviimp	51	67.0%	94.7%	99.8%	91.0%	98.1%	92.1%
		930	—	—	—	—	930
		100.0%	—	—	—	—	100.0%
		1.0%	—	—	—	—	0.2%
%	52	68.0%	94.7%	99.8%	91.0%	98.1%	92.3%
		835	—	—	—	72	907
		92.1%	—	—	—	7.9%	100.0%
		0.9%	—	—	—	0.0%	0.2%
lpq	53	68.9%	94.7%	99.8%	91.0%	98.1%	92.5%
		774	59	—	—	1	834
		92.8%	7.1%	—	—	0.1%	100.0%
		0.9%	0.1%	—	—	0.0%	0.2%
dirs	54	69.8%	94.7%	99.8%	91.0%	98.1%	92.6%
		827	—	—	—	—	827
		100.0%	—	—	—	—	100.0%
		0.9%	—	—	—	—	0.2%
f	55	70.7%	94.7%	99.8%	91.0%	98.1%	92.8%

Table 23 - One Hundred Commands from Each Source (6)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		761	—	—	—	—	761
		100.0%	—	—	—	—	100.0%
		0.8%	—	—	—	—	0.2%
ipq	56	71.5%	94.7%	99.8%	91.0%	98.1%	93.0%
		288	353	—	10	55	706
		40.8%	50.0%	—	1.4%	7.8%	100.0%
		0.3%	0.6%	—	0.2%	0.0%	0.1%
uptime	57	71.9%	95.3%	99.8%	91.2%	98.1%	93.1%
		517	171	—	—	7	695
		74.4%	24.6%	—	—	1.0%	100.0%
		0.6%	0.3%	—	—	0.0%	0.1%
from	58	72.4%	95.6%	99.8%	91.2%	98.1%	93.3%
		546	68	—	46	3	663
		82.4%	10.3%	—	6.9%	0.5%	100.0%
		0.6%	0.1%	—	1.0%	0.0%	0.1%
who	59	73.0%	95.7%	99.8%	92.3%	98.1%	93.4%
		409	—	—	—	245	654
		62.5%	—	—	—	37.5%	100.0%
		0.5%	—	—	—	0.1%	0.1%
lpr	60	73.5%	95.7%	99.8%	92.3%	98.3%	93.5%
		627	—	—	—	7	634
		98.9%	—	—	—	1.1%	100.0%
		0.7%	—	—	—	0.0%	0.1%
make	61	74.2%	95.7%	99.8%	92.3%	98.3%	93.7%
		—	—	—	—	632	632
		—	—	—	—	100.0%	100.0%
		—	—	—	—	0.3%	0.1%
$troff	62	74.2%	95.7%	99.8%	92.3%	98.6%	93.8%
		606	—	—	—	12	618
		98.1%	—	—	—	1.9%	100.0%
		0.7%	—	—	—	0.0%	0.1%
rwho	63	74.8%	95.7%	99.8%	92.3%	98.6%	93.9%
		—	—	—	—	609	609
		—	—	—	—	100.0%	100.0%
		—	—	—	—	0.3%	0.1%
$lpr	64	74.8%	95.7%	99.8%	92.3%	98.9%	94.1%
		540	—	—	—	65	605
		89.3%	—	—	—	10.7%	100.0%
		0.6%	—	—	—	0.0%	0.1%
cp	65	75.4%	95.7%	99.8%	92.3%	98.9%	94.2%
		564	—	—	—	39	603
		93.5%	—	—	—	6.5%	100.0%
		0.6%	—	—	—	0.0%	0.1%
pushd	66	76.1%	95.7%	99.8%	92.3%	98.9%	94.3%

Table 23 - One Hundred Commands from Each Source (7)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		602	—	—	—	—	602
		100.0%	—	—	—	—	100.0%
		0.7%	—	—	—	—	0.1%
pi	67	76.7%	95.7%	99.8%	92.3%	98.9%	94.4%
		576	—	—	—	—	576
		100.0%	—	—	—	—	100.0%
		0.6%	—	—	—	—	0.1%
man	68	77.4%	95.7%	99.8%	92.3%	98.9%	94.6%
		466	—	—	—	78	544
		85.7%	—	—	—	14.3%	100.0%
		0.5%	—	—	—	0.0%	0.1%
ditroff	69	77.9%	95.7%	99.8%	92.3%	98.9%	94.7%
		498	21	—	—	1	520
		95.8%	4.0%	—	—	0.2%	100.0%
		0.5%	0.0%	—	—	0.0%	0.1%
egrep	70	78.4%	95.8%	99.8%	92.3%	98.9%	94.8%
		57	448	1	2	11	519
		11.0%	86.3%	0.2%	0.4%	2.1%	100.0%
		0.1%	0.7%	0.0%	0.0%	0.0%	0.1%
biff	71	78.5%	96.5%	99.8%	92.3%	98.9%	94.9%
		475	—	—	—	33	508
		93.5%	—	—	—	6.5%	100.0%
		0.5%	—	—	—	0.0%	0.1%
rcp	72	79.0%	96.5%	99.8%	92.3%	99.0%	95.0%
		453	50	—	—	2	505
		89.7%	9.9%	—	—	0.4%	100.0%
		0.5%	0.1%	—	—	0.0%	0.1%
df	73	79.5%	96.6%	99.8%	92.3%	99.0%	95.1%
		473	10	7	1	9	500
		94.6%	2.0%	1.4%	0.2%	1.8%	100.0%
		0.5%	0.0%	0.0%	0.0%	0.0%	0.1%
chdir	74	80.0%	96.6%	99.8%	92.4%	99.0%	95.2%
		13	101	—	—	385	499
		2.6%	20.2%	—	—	77.2%	100.0%
		0.0%	0.2%	—	—	0.2%	0.1%
case	75	80.0%	96.8%	99.8%	92.4%	99.1%	95.3%
		497	—	—	—	—	497
		100.0%	—	—	—	—	100.0%
		0.5%	—	—	—	—	0.1%
bg	76	80.6%	96.8%	99.8%	92.4%	99.1%	95.4%
		459	14	—	20	1	494
		92.9%	2.8%	—	4.0%	0.2%	100.0%
		0.5%	0.0%	—	0.5%	0.0%	0.1%
kill	77	81.1%	96.8%	99.8%	92.8%	99.1%	95.5%

Table 23 - One Hundred Commands from Each Source (8)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		439	17	—	—	13	469
		93.6%	3.6%	—	—	2.8%	100.0%
		0.5%	0.0%	—	—	0.0%	0.1%
Mail	78	81.6%	96.8%	99.8%	92.8%	99.2%	95.6%
		—	80	25	42	314	461
		—	17.4%	5.4%	9.1%	68.1%	100.0%
		—	0.1%	0.0%	1.0%	0.1%	0.1%
breaksw	79	81.6%	96.9%	99.8%	93.8%	99.3%	95.7%
		36	18	156	—	218	428
		8.4%	4.2%	36.4%	—	50.9%	100.0%
		0.0%	0.0%	0.1%	—	0.1%	0.1%
foreach	80	81.6%	97.0%	100.0%	93.8%	99.4%	95.8%
		335	55	—	—	—	390
		85.9%	14.1%	—	—	—	100.0%
		0.4%	0.1%	—	—	—	0.1%
w	81	82.0%	97.1%	100.0%	93.8%	99.4%	95.9%
		5	338	—	—	11	354
		1.4%	95.5%	—	—	3.1%	100.0%
		0.0%	0.6%	—	—	0.0%	0.1%
sysline	82	82.0%	97.6%	100.0%	93.8%	99.4%	96.0%
		344	—	—	9	—	353
		97.5%	—	—	2.5%	—	100.0%
		0.4%	—	—	0.2%	—	0.1%
history	83	82.4%	97.6%	100.0%	94.0%	99.4%	96.0%
		331	21	—	—	—	352
		94.0%	6.0%	—	—	—	100.0%
		0.4%	0.0%	—	—	—	0.1%
rwhoson	84	82.7%	97.7%	100.0%	94.0%	99.4%	96.1%
		335	—	—	—	—	335
		100.0%	—	—	—	—	100.0%
		0.4%	—	—	—	—	0.1%
talk	85	83.1%	97.7%	100.0%	94.0%	99.4%	96.2%
		332	—	—	—	—	332
		100.0%	—	—	—	—	100.0%
		0.4%	—	—	—	—	0.1%
obj	86	83.5%	97.7%	100.0%	94.0%	99.4%	96.2%
		202	119	—	—	4	325
		62.2%	36.6%	—	—	1.2%	100.0%
		0.2%	0.2%	—	—	0.0%	0.1%
mc	87	83.7%	97.8%	100.0%	94.0%	99.4%	96.3%
		323	—	—	—	—	323
		100.0%	—	—	—	—	100.0%
		0.4%	—	—	—	—	0.1%
sccs	88	84.1%	97.8%	100.0%	94.0%	99.4%	96.4%

Table 23 - One Hundred Commands from Each Source (9)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		316	—	—	—	—	316
		100.0%	—	—	—	—	100.0%
		0.3%	—	—	—	—	0.1%
0	89	84.4%	97.8%	100.0%	94.0%	99.4%	96.4%
		286	—	—	—	3	289
		99.0%	—	—	—	1.0%	100.0%
		0.3%	—	—	—	0.0%	0.1%
cc	90	84.7%	97.8%	100.0%	94.0%	99.4%	96.5%
		199	77	—	1	4	281
		70.8%	27.4%	—	0.4%	1.4%	100.0%
		0.2%	0.1%	—	0.0%	0.0%	0.1%
mkdir	91	84.9%	98.0%	100.0%	94.0%	99.4%	96.6%
		259	—	—	—	—	259
		100.0%	—	—	—	—	100.0%
		0.3%	—	—	—	—	0.1%
tail	92	85.2%	98.0%	100.0%	94.0%	99.4%	96.6%
		200	7	—	—	50	257
		77.8%	2.7%	—	—	19.5%	100.0%
		0.2%	0.0%	—	—	0.0%	0.1%
chmod	93	85.4%	98.0%	100.0%	94.0%	99.4%	96.7%
		214	29	2	7	3	255
		83.9%	11.4%	0.8%	2.7%	1.2%	100.0%
		0.2%	0.0%	0.0%	0.2%	0.0%	0.1%
fgrep	94	85.7%	98.0%	100.0%	94.1%	99.4%	96.7%
		255	—	—	—	—	255
		100.0%	—	—	—	—	100.0%
		0.3%	—	—	—	—	0.1%
ln	95	86.0%	98.0%	100.0%	94.1%	99.4%	96.8%
		6	—	—	—	242	248
		2.4%	—	—	—	97.6%	100.0%
		0.0%	—	—	—	0.1%	0.1%
troff	96	86.0%	98.0%	100.0%	94.1%	99.6%	96.8%
		18	219	—	—	6	243
		7.4%	90.1%	—	—	2.5%	100.0%
		0.0%	0.4%	—	—	0.0%	0.1%
mesg	97	86.0%	98.4%	100.0%	94.1%	99.6%	96.9%
		201	—	—	—	26	227
		88.5%	—	—	—	11.5%	100.0%
		0.2%	—	—	—	0.0%	0.0%
page	98	86.2%	98.4%	100.0%	94.1%	99.6%	96.9%
		224	—	—	—	—	224
		100.0%	—	—	—	—	100.0%
		0.2%	—	—	—	—	0.0%
tar	99	86.4%	98.4%	100.0%	94.1%	99.6%	97.0%

Table 23 - One Hundred Commands from Each Source (10)							
Command	Rank	Terminal	.login	.cshrc	.logout	Command File	Totals
		202	—	—	—	20	222
		91.0%	—	—	—	9.0%	100.0%
		0.2%	—	—	—	0.0%	0.0%
itroff	100	86.7%	98.4%	100.0%	94.1%	99.6%	97.0%
		90774	60796	110996	4393	211577	478536
Totals		19.0%	12.7%	23.2%	0.9%	44.2%	100.0%
for all		100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Commands		100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

Table 24 - One Hundred Commands from Each Class (1)

Command	Rank	Secretary	Facu	Grad	Guest	Misc-Human	Misc Non-Human	Prog	Root	Unknown	Totals
		9196	13712	36535	11406	15037	—	16752	4167	—	106805
		8.6%	12.8%	34.2%	10.7%	14.1%	—	15.7%	3.9%	—	100.0%
		29.0%	22.0%	36.8%	26.8%	41.1%	—	31.0%	35.7%	—	22.3%
alias	1	29.0%	22.0%	36.8%	26.8%	41.1%	—	31.0%	35.7%	—	22.3%
		3957	10619	12685	5008	3462	18484	6432	1695	—	62342
		6.3%	17.0%	20.3%	8.0%	5.6%	29.6%	10.3%	2.7%	—	100.0%
		12.5%	17.0%	12.8%	11.8%	9.5%	13.2%	11.9%	14.5%	—	13.0%
set	2	41.5%	39.0%	49.6%	38.5%	50.5%	13.2%	42.8%	50.2%	—	35.3%
		1953	6314	4718	2597	823	28324	3256	100	—	48085
		4.1%	13.1%	9.8%	5.4%	1.7%	58.9%	6.8%	0.2%	—	100.0%
		6.2%	10.1%	4.8%	6.1%	2.2%	20.2%	6.0%	0.9%	—	10.0%
if	3	47.6%	49.2%	54.4%	44.6%	52.8%	33.4%	48.9%	51.1%	—	45.4%
		370	1389	731	118	12	21350	119	—	—	24089
		1.5%	5.8%	3.0%	0.5%	0.0%	88.6%	0.5%	—	—	100.0%
		1.2%	2.2%	0.7%	0.3%	0.0%	15.2%	0.2%	—	—	5.0%
shift	4	48.8%	51.4%	55.1%	44.9%	52.8%	48.6%	49.1%	51.1%	—	50.4%
		494	1658	882	228	185	15374	188	—	—	19009
		2.6%	8.7%	4.6%	1.2%	1.0%	80.9%	1.0%	—	—	100.0%
		1.6%	2.7%	0.9%	0.5%	0.5%	11.0%	0.3%	—	—	4.0%
switch	5	50.4%	54.1%	56.0%	45.5%	53.3%	59.5%	49.4%	51.1%	—	54.4%
		370	1390	739	110	12	15373	121	—	—	18115
		2.0%	7.7%	4.1%	0.6%	0.1%	84.9%	0.7%	—	—	100.0%
		1.2%	2.2%	0.7%	0.3%	0.0%	11.0%	0.2%	—	—	3.8%
goto	6	51.5%	56.3%	56.7%	45.7%	53.3%	70.5%	49.6%	51.1%	—	58.2%
		311	3555	1977	1341	1840	5383	467	8	—	14882
		2.1%	23.9%	13.3%	9.0%	12.4%	36.2%	3.1%	0.1%	—	100.0%
		1.0%	5.7%	2.0%	3.1%	5.0%	3.8%	0.9%	0.1%	—	3.1%
echo	7	52.5%	62.0%	58.7%	48.9%	58.4%	74.3%	50.5%	51.2%	—	61.3%
		1248	1429	4029	3033	1439	—	2295	8	—	13481
		9.3%	10.6%	29.9%	22.5%	10.7%	—	17.0%	0.1%	—	100.0%
		3.9%	2.3%	4.1%	7.1%	3.9%	—	4.2%	0.1%	—	2.8%
setenv	8	56.4%	64.3%	62.8%	56.0%	62.3%	74.3%	54.8%	51.2%	—	64.1%
		1063	2432	3158	1369	1564	—	1232	409	—	11227
		9.5%	21.7%	28.1%	12.2%	13.9%	—	11.0%	3.6%	—	100.0%
		3.4%	3.9%	3.2%	3.2%	4.3%	—	2.3%	3.5%	—	2.3%
ls	9	59.8%	68.2%	66.0%	59.2%	66.6%	74.3%	57.0%	54.7%	—	66.5%
		810	1296	1184	722	205	5979	575	93	—	10864
		7.5%	11.9%	10.9%	6.6%	1.9%	55.0%	5.3%	0.9%	—	100.0%
		2.6%	2.1%	1.2%	1.7%	0.6%	4.3%	1.1%	0.8%	—	2.3%
endif	10	62.4%	70.3%	67.2%	60.9%	67.1%	78.6%	58.1%	55.5%	—	68.7%
		151	144	305	182	133	6240	118	64	—	7337
		2.1%	2.0%	4.2%	2.5%	1.8%	85.0%	1.6%	0.9%	—	100.0%
		0.5%	0.2%	0.3%	0.4%	0.4%	4.4%	0.2%	0.5%	—	1.5%
exit	11	62.8%	70.5%	67.5%	61.3%	67.5%	83.0%	58.3%	56.1%	—	70.3%
		80	817	1182	98	696	611	2595	779	—	6858
		1.2%	11.9%	17.2%	1.4%	10.1%	8.9%	37.8%	11.4%	—	100.0%
		0.3%	1.3%	1.2%	0.2%	1.9%	0.4%	4.8%	6.7%	—	1.4%
end	12	63.1%	71.8%	68.7%	61.6%	69.4%	83.5%	63.1%	62.7%	—	71.7%
		290	417	523	256	223	2952	250	820	—	5731
		5.1%	7.3%	9.1%	4.5%	3.9%	51.5%	4.4%	14.3%	—	100.0%
		0.9%	0.7%	0.5%	0.6%	0.6%	2.1%	0.5%	7.0%	—	1.2%
rm	13	64.0%	72.5%	69.2%	62.2%	70.0%	85.6%	63.6%	69.8%	—	72.9%
		698	734	1635	671	929	26	630	315	—	5638
		12.4%	13.0%	29.0%	11.9%	16.5%	0.5%	11.2%	5.6%	—	100.0%
		2.2%	1.2%	1.6%	1.6%	2.5%	0.0%	1.2%	2.7%	—	1.2%
cd	14	66.2%	73.7%	70.8%	63.7%	72.5%	85.6%	64.7%	72.5%	—	74.1%
		351	581	572	321	147	2991	409	—	—	5372
		6.5%	10.8%	10.6%	6.0%	2.7%	55.7%	7.6%	—	—	100.0%
		1.1%	0.9%	0.6%	0.8%	0.4%	2.1%	0.8%	—	—	1.1%
umask	15	67.3%	74.6%	71.4%	64.5%	72.9%	87.7%	65.5%	72.5%	—	75.2%
		735	1146	1967	232	686	—	389	125	—	5280
		13.9%	21.7%	37.3%	4.4%	13.0%	—	7.4%	2.4%	—	100.0%
		2.3%	1.8%	2.0%	0.5%	1.9%	—	0.7%	1.1%	—	1.1%
vi	16	69.6%	76.4%	73.4%	65.0%	74.8%	87.7%	66.2%	73.5%	—	76.3%

Table 24 - One Hundred Commands from Each Class (2)

Command	Rank	Secretary	Facu	Grad	Guest	Misc-Human	Misc Non-Human	Prog	Root	Unknown	Totals
		92	546	322	265	61	2812	297	707	—	5102
		1.8%	10.7%	6.3%	5.2%	1.2%	55.1%	5.8%	13.9%	—	100.0%
		0.3%	0.9%	0.3%	0.6%	0.2%	2.0%	0.5%	6.1%	—	1.1%
cat	17	69.9%	77.3%	73.7%	65.7%	75.0%	89.7%	66.8%	79.6%	—	77.4%
		521	806	1289	309	585	—	717	162	—	4389
		11.9%	18.4%	29.4%	7.0%	13.3%	—	16.3%	3.7%	—	100.0%
		1.6%	1.3%	1.3%	0.7%	1.6%	—	1.3%	1.4%	—	0.9%
more	18	71.6%	78.6%	75.0%	66.4%	76.6%	89.7%	68.1%	81.0%	—	78.3%
		242	457	303	110	61	2991	73	—	—	4237
		5.7%	10.8%	7.2%	2.6%	1.4%	70.6%	1.7%	—	—	100.0%
		0.8%	0.7%	0.3%	0.3%	0.2%	2.1%	0.1%	—	—	0.9%
LABEL	19	72.3%	79.3%	75.3%	66.6%	76.7%	91.9%	68.2%	81.0%	—	79.2%
		182	499	1090	1036	338	—	828	36	—	4009
		4.5%	12.4%	27.2%	25.8%	8.4%	—	20.7%	0.9%	—	100.0%
		0.6%	0.8%	1.1%	2.4%	0.9%	—	1.5%	0.3%	—	0.8%
jobs	20	72.9%	80.1%	76.4%	69.1%	77.7%	91.9%	69.7%	81.3%	—	80.0%
		360	681	1140	681	267	—	552	2	—	3683
		9.8%	18.5%	31.0%	18.5%	7.2%	—	15.0%	0.1%	—	100.0%
		1.1%	1.1%	1.1%	1.6%	0.7%	—	1.0%	0.0%	—	0.8%
unset	21	74.0%	81.2%	77.6%	70.7%	78.4%	91.9%	70.8%	81.3%	—	80.8%
		49	664	261	56	19	585	1784	—	—	3418
		1.4%	19.4%	7.6%	1.6%	0.6%	17.1%	52.2%	—	—	100.0%
		0.2%	1.1%	0.3%	0.1%	0.1%	0.4%	3.3%	—	—	0.7%
while	22	74.2%	82.3%	77.8%	70.8%	78.4%	92.3%	74.1%	81.3%	—	81.5%
		275	313	928	583	600	—	546	2	—	3247
		8.5%	9.6%	28.6%	18.0%	18.5%	—	16.8%	0.1%	—	100.0%
		0.9%	0.5%	0.9%	1.4%	1.6%	—	1.0%	0.0%	—	0.7%
stty	23	75.0%	82.8%	78.8%	72.2%	80.1%	92.3%	75.1%	81.3%	—	82.2%
		49	137	3	66	12	1092	1723	—	—	3082
		1.6%	4.4%	0.1%	2.1%	0.4%	35.4%	55.9%	—	—	100.0%
		0.2%	0.2%	0.0%	0.2%	0.0%	0.8%	3.2%	—	—	0.6%
@	24	75.2%	83.0%	78.8%	72.3%	80.1%	93.1%	78.3%	81.3%	—	82.8%
		280	700	666	652	167	—	414	9	—	2888
		9.7%	24.2%	23.1%	22.6%	5.8%	—	14.3%	0.3%	—	100.0%
		0.9%	1.1%	0.7%	1.5%	0.5%	—	0.8%	0.1%	—	0.6%
fg	25	76.1%	84.1%	79.4%	73.9%	80.6%	93.1%	79.0%	81.4%	—	83.4%
		—	—	1	—	—	2812	—	—	—	2813
		—	—	0.0%	—	—	100.0%	—	—	—	100.0%
		—	—	0.0%	—	—	2.0%	—	—	—	0.6%
ips	26	76.1%	84.1%	79.4%	73.9%	80.6%	95.1%	79.0%	81.4%	—	84.0%
		—	—	10	74	—	2567	—	—	—	2651
		—	—	0.4%	2.8%	—	96.8%	—	—	—	100.0%
		—	—	0.0%	0.2%	—	1.8%	—	—	—	0.6%
onintr	27	76.1%	84.1%	79.5%	74.0%	80.6%	96.9%	79.0%	81.4%	—	84.6%
		283	370	734	403	156	—	416	—	—	2362
		12.0%	15.7%	31.1%	17.1%	6.6%	—	17.6%	—	—	100.0%
		0.9%	0.6%	0.7%	0.9%	0.4%	—	0.8%	—	—	0.5%
eval	28	77.0%	84.7%	80.2%	75.0%	81.0%	96.9%	79.8%	81.4%	—	85.1%
		717	241	643	390	129	—	233	—	—	2353
		30.5%	10.2%	27.3%	16.6%	5.5%	—	9.9%	—	—	100.0%
		2.3%	0.4%	0.6%	0.9%	0.4%	—	0.4%	—	—	0.5%
mail	29	79.2%	85.1%	80.8%	75.9%	81.3%	96.9%	80.2%	81.4%	—	85.5%
		231	85	1199	224	166	—	439	2	—	2346
		9.8%	3.6%	51.1%	9.5%	7.1%	—	18.7%	0.1%	—	100.0%
		0.7%	0.1%	1.2%	0.5%	0.5%	—	0.8%	0.0%	—	0.5%
source	30	80.0%	85.3%	82.1%	76.4%	81.8%	96.9%	81.0%	81.4%	—	86.0%
		265	209	817	289	125	—	525	35	—	2265
		11.7%	9.2%	36.1%	12.8%	5.5%	—	23.2%	1.5%	—	100.0%
		0.8%	0.3%	0.8%	0.7%	0.3%	—	1.0%	0.3%	—	0.5%
clear	31	80.8%	85.6%	82.9%	77.1%	82.1%	96.9%	82.0%	81.7%	—	86.5%
		322	149	683	486	296	—	244	—	—	2180
		14.8%	6.8%	31.3%	22.3%	13.6%	—	11.2%	—	—	100.0%
		1.0%	0.2%	0.7%	1.1%	0.8%	—	0.5%	—	—	0.5%
mags	32	81.8%	85.8%	83.6%	78.2%	82.9%	96.9%	82.4%	81.7%	—	87.0%

Table 24 - One Hundred Commands from Each Class (3)

Command	Rank	Secretary	Facu	Grad	Guest	Misc-Human	Misc Non-Human	Prog	Root	Unknown	Totals
		—	51	156	13	—	—	1650	—	—	1870
		—	2.7%	8.3%	0.7%	—	—	88.2%	—	—	100.0%
		—	0.1%	0.2%	0.0%	—	—	3.0%	—	—	0.4%
sleep	33	81.8%	85.9%	83.7%	78.3%	82.9%	96.9%	85.5%	81.7%	—	87.4%
		94	1052	185	247	193	—	66	—	—	1837
		5.1%	57.3%	10.1%	13.4%	10.5%	—	3.6%	—	—	100.0%
		0.3%	1.7%	0.2%	0.6%	0.5%	—	0.1%	—	—	0.4%
else	34	82.1%	87.6%	83.9%	78.8%	83.5%	96.9%	85.6%	81.7%	—	87.7%
		120	297	896	170	152	—	59	47	—	1741
		6.9%	17.1%	51.5%	9.8%	8.7%	—	3.4%	2.7%	—	100.0%
		0.4%	0.5%	0.9%	0.4%	0.4%	—	0.1%	0.4%	—	0.4%
ps	35	82.5%	88.1%	84.8%	79.2%	83.9%	96.9%	85.7%	82.1%	—	88.1%
		92	338	316	37	184	624	67	36	—	1694
		5.4%	20.0%	18.7%	2.2%	10.9%	36.8%	4.0%	2.1%	—	100.0%
		0.3%	0.5%	0.3%	0.1%	0.5%	0.4%	0.1%	0.3%	—	0.4%
mv	36	82.8%	88.6%	85.1%	79.3%	84.4%	97.3%	85.9%	82.4%	—	88.5%
		—	—	—	—	—	1483	—	—	—	1483
		—	—	—	—	—	100.0%	—	—	—	100.0%
		—	—	—	—	—	1.1%	—	—	—	0.3%
dit-imagen	37	82.8%	88.6%	85.1%	79.3%	84.4%	98.4%	85.9%	82.4%	—	88.8%
		92	64	389	172	277	—	388	10	—	1392
		6.6%	4.6%	27.9%	12.4%	19.9%	—	27.9%	0.7%	—	100.0%
		0.3%	0.1%	0.4%	0.4%	0.8%	—	0.7%	0.1%	—	0.3%
rlogin	38	83.1%	88.7%	85.5%	79.7%	85.1%	98.4%	86.6%	82.5%	—	89.1%
		212	42	461	233	112	—	289	6	—	1355
		15.6%	3.1%	34.0%	17.2%	8.3%	—	21.3%	0.4%	—	100.0%
		0.7%	0.1%	0.5%	0.5%	0.3%	—	0.5%	0.1%	—	0.3%
ls	39	83.7%	88.8%	86.0%	80.3%	85.4%	98.4%	87.1%	82.6%	—	89.3%
		164	58	193	96	259	—	184	325	—	1279
		12.8%	4.5%	15.1%	7.5%	20.3%	—	14.4%	25.4%	—	100.0%
		0.5%	0.1%	0.2%	0.2%	0.7%	—	0.3%	2.8%	—	0.3%
date	40	84.3%	88.9%	86.2%	80.5%	86.2%	98.4%	87.4%	85.4%	—	89.6%
		100	114	397	172	202	—	238	—	—	1223
		8.2%	9.3%	32.5%	14.1%	16.5%	—	19.5%	—	—	100.0%
		0.3%	0.2%	0.4%	0.4%	0.6%	—	0.4%	—	—	0.3%
logout	41	84.6%	89.1%	86.6%	80.9%	86.7%	98.4%	87.9%	85.4%	—	89.9%
		57	190	580	280	4	—	57	4	—	1172
		4.9%	16.2%	49.5%	23.9%	0.3%	—	4.9%	0.3%	—	100.0%
		0.2%	0.3%	0.6%	0.7%	0.0%	—	0.1%	0.0%	—	0.2%
u	42	84.8%	89.4%	87.2%	81.6%	86.7%	98.4%	88.0%	85.4%	—	90.1%
		36	131	268	112	84	—	209	323	—	1163
		3.1%	11.3%	23.0%	9.6%	7.2%	—	18.0%	27.8%	—	100.0%
		0.1%	0.2%	0.3%	0.3%	0.2%	—	0.4%	2.8%	—	0.2%
ruptime	43	84.9%	89.6%	87.4%	81.8%	86.9%	98.4%	88.4%	88.2%	—	90.3%
		114	22	557	75	51	—	224	114	—	1157
		9.9%	1.9%	48.1%	6.5%	4.4%	—	19.4%	9.9%	—	100.0%
		0.4%	0.0%	0.6%	0.2%	0.1%	—	0.4%	1.0%	—	0.2%
pwd	44	85.2%	89.6%	88.0%	82.0%	87.1%	98.4%	88.8%	89.1%	—	90.6%
		163	40	260	222	95	—	327	41	—	1148
		14.2%	3.5%	22.6%	19.3%	8.3%	—	28.5%	3.6%	—	100.0%
		0.5%	0.1%	0.3%	0.5%	0.3%	—	0.6%	0.4%	—	0.2%
finger	45	85.7%	89.7%	88.3%	82.5%	87.3%	98.4%	89.4%	89.5%	—	90.8%
		219	174	340	39	108	—	225	—	—	1105
		19.8%	15.7%	30.8%	3.5%	9.8%	—	20.4%	—	—	100.0%
		0.7%	0.3%	0.3%	0.1%	0.3%	—	0.4%	—	—	0.2%
fortune	46	86.4%	90.0%	88.6%	82.6%	87.6%	98.4%	89.8%	89.5%	—	91.1%
		58	275	306	77	23	—	221	49	—	1009
		5.7%	27.3%	30.3%	7.6%	2.3%	—	21.9%	4.9%	—	100.0%
		0.2%	0.4%	0.3%	0.2%	0.1%	—	0.4%	0.4%	—	0.2%
grep	47	86.6%	90.4%	88.9%	82.8%	87.7%	98.4%	90.2%	89.9%	—	91.3%
		—	—	—	996	—	—	—	—	—	996
		—	—	—	100.0%	—	—	—	—	—	100.0%
		—	—	—	2.3%	—	—	—	—	—	0.2%
4	48	86.6%	90.4%	88.9%	85.1%	87.7%	98.4%	90.2%	89.9%	—	91.5%

Table 24 - One Hundred Commands from Each Class (4)

Command	Rank	Secretary	Facu	Grad	Guest	Misc-Human	Misc Non-Human	Prog	Root	Unknown	Totals
		—	—	—	—	—	963	—	—	—	963
		—	—	—	—	—	100.0%	—	—	—	100.0%
		—	—	—	—	—	0.7%	—	—	—	0.2%
catdvi	49	86.6%	90.4%	88.9%	85.1%	87.7%	99.1%	90.2%	89.9%	—	91.7%
		88	65	221	185	152	—	217	25	—	953
		9.2%	6.8%	23.2%	19.4%	15.9%	—	22.8%	2.6%	—	100.0%
		0.3%	0.1%	0.2%	0.4%	0.4%	—	0.4%	0.2%	—	0.2%
rsh	50	86.9%	90.5%	89.1%	85.6%	88.1%	99.1%	90.6%	90.1%	—	91.9%
		—	—	—	—	2	946	—	—	—	948
		—	—	—	—	0.2%	99.8%	—	—	—	100.0%
		—	—	—	—	0.0%	0.7%	—	—	—	0.2%
dviimp	51	86.9%	90.5%	89.1%	85.6%	88.1%	99.8%	90.6%	90.1%	—	92.1%
		51	69	176	261	64	—	307	2	—	930
		5.5%	7.4%	18.9%	28.1%	6.9%	—	33.0%	0.2%	—	100.0%
		0.2%	0.1%	0.2%	0.6%	0.2%	—	0.6%	0.0%	—	0.2%
%	52	87.1%	90.6%	89.3%	86.2%	88.3%	99.8%	91.2%	90.1%	—	92.3%
		121	256	314	127	41	—	36	12	—	907
		13.3%	28.2%	34.6%	14.0%	4.5%	—	4.0%	1.3%	—	100.0%
		0.4%	0.4%	0.3%	0.3%	0.1%	—	0.1%	0.1%	—	0.2%
lpq	53	87.4%	91.0%	89.6%	86.5%	88.4%	99.8%	91.3%	90.2%	—	92.5%
		1	19	272	386	120	—	35	1	—	834
		0.1%	2.3%	32.6%	46.3%	14.4%	—	4.2%	0.1%	—	100.0%
		0.0%	0.0%	0.3%	0.9%	0.3%	—	0.1%	0.0%	—	0.2%
dirs	54	87.4%	91.1%	89.9%	87.4%	88.7%	99.8%	91.3%	90.2%	—	92.6%
		7	95	270	254	144	—	51	6	—	827
		0.8%	11.5%	32.6%	30.7%	17.4%	—	6.2%	0.7%	—	100.0%
		0.0%	0.2%	0.3%	0.6%	0.4%	—	0.1%	0.1%	—	0.2%
f	55	87.5%	91.2%	90.2%	88.0%	89.1%	99.8%	91.4%	90.3%	—	92.8%
		53	257	325	—	16	—	99	11	—	761
		7.0%	33.8%	42.7%	—	2.1%	—	13.0%	1.4%	—	100.0%
		0.2%	0.4%	0.3%	—	0.0%	—	0.2%	0.1%	—	0.2%
ipq	56	87.6%	91.6%	90.5%	88.0%	89.2%	99.8%	91.6%	90.4%	—	93.0%
		116	50	336	102	52	—	50	—	—	706
		16.4%	7.1%	47.6%	14.4%	7.4%	—	7.1%	—	—	100.0%
		0.4%	0.1%	0.3%	0.2%	0.1%	—	0.1%	—	—	0.1%
uptime	57	88.0%	91.7%	90.8%	88.2%	89.3%	99.8%	91.7%	90.4%	—	93.1%
		156	21	51	194	62	—	210	1	—	695
		22.4%	3.0%	7.3%	27.9%	8.9%	—	30.2%	0.1%	—	100.0%
		0.5%	0.0%	0.1%	0.5%	0.2%	—	0.4%	0.0%	—	0.1%
from	58	88.5%	91.7%	90.9%	88.7%	89.5%	99.8%	92.1%	90.4%	—	93.3%
		38	22	204	125	163	—	98	13	—	663
		5.7%	3.3%	30.8%	18.9%	24.6%	—	14.8%	2.0%	—	100.0%
		0.1%	0.0%	0.2%	0.3%	0.4%	—	0.2%	0.1%	—	0.1%
who	59	88.6%	91.8%	91.1%	89.0%	89.9%	99.8%	92.3%	90.5%	—	93.4%
		163	149	146	26	55	—	107	8	—	654
		24.9%	22.8%	22.3%	4.0%	8.4%	—	16.4%	1.2%	—	100.0%
		0.5%	0.2%	0.1%	0.1%	0.2%	—	0.2%	0.1%	—	0.1%
lpr	60	89.1%	92.0%	91.2%	89.0%	90.1%	99.8%	92.5%	90.6%	—	93.5%
		9	181	325	55	54	—	5	5	—	634
		1.4%	28.5%	51.3%	8.7%	8.5%	—	0.8%	0.8%	—	100.0%
		0.0%	0.3%	0.3%	0.1%	0.1%	—	0.0%	0.0%	—	0.1%
make	61	89.1%	92.3%	91.6%	89.2%	90.2%	99.8%	92.5%	90.6%	—	93.7%
		68	288	201	36	4	—	35	—	—	632
		10.8%	45.6%	31.8%	5.7%	0.6%	—	5.5%	—	—	100.0%
		0.2%	0.5%	0.2%	0.1%	0.0%	—	0.1%	—	—	0.1%
$troff	62	89.4%	92.8%	91.8%	89.3%	90.2%	99.8%	92.5%	90.6%	—	93.8%
		10	51	210	141	58	—	144	4	—	618
		1.6%	8.3%	34.0%	22.8%	9.4%	—	23.3%	0.6%	—	100.0%
		0.0%	0.1%	0.2%	0.3%	0.2%	—	0.3%	0.0%	—	0.1%
rwho	63	89.4%	92.8%	92.0%	89.6%	90.4%	99.8%	92.8%	90.7%	—	93.9%
		68	286	182	34	4	—	35	—	—	609
		11.2%	47.0%	29.9%	5.6%	0.7%	—	5.7%	—	—	100.0%
		0.2%	0.5%	0.2%	0.1%	0.0%	—	0.1%	—	—	0.1%
$lpr	64	89.6%	93.3%	92.2%	89.7%	90.4%	99.8%	92.9%	90.7%	—	94.1%

Table 24 - One Hundred Commands from Each Class (5)

Command	Rank	Secretary	Facu	Grad	Guest	Misc-Human	Misc Non-Human	Prog	Root	Unknown	Totals
		31	87	179	48	157	39	22	42	—	605
		5.1%	14.4%	29.6%	7.9%	26.0%	6.4%	3.6%	6.9%	—	100.0%
		0.1%	0.1%	0.2%	0.1%	0.4%	0.0%	0.0%	0.4%	—	0.1%
cp	65	89.7%	93.4%	92.3%	89.8%	90.8%	99.8%	92.9%	91.0%	—	94.2%
		16	173	273	70	35	—	36	—	—	603
		2.7%	28.7%	45.3%	11.6%	5.8%	—	6.0%	—	—	100.0%
		0.1%	0.3%	0.3%	0.2%	0.1%	—	0.1%	—	—	0.1%
pushd	66	89.8%	93.7%	92.6%	89.9%	90.9%	99.8%	93.0%	91.0%	—	94.3%
		21	16	251	—	288	—	26	—	—	602
		3.5%	2.7%	41.7%	—	47.8%	—	4.3%	—	—	100.0%
		0.1%	0.0%	0.3%	—	0.8%	—	0.0%	—	—	0.1%
pi	67	89.8%	93.8%	92.9%	89.9%	91.7%	99.8%	93.0%	91.0%	—	94.4%
		31	96	185	87	71	—	88	18	—	576
		5.4%	16.7%	32.1%	15.1%	12.3%	—	15.3%	3.1%	—	100.0%
		0.1%	0.2%	0.2%	0.2%	0.2%	—	0.2%	0.2%	—	0.1%
man	68	89.9%	93.9%	93.1%	90.1%	91.9%	99.8%	93.2%	91.2%	—	94.6%
		88	233	162	28	4	—	29	—	—	544
		16.2%	42.8%	29.8%	5.1%	0.7%	—	5.3%	—	—	100.0%
		0.3%	0.4%	0.2%	0.1%	0.0%	—	0.1%	—	—	0.1%
ditroff	69	90.2%	94.3%	93.2%	90.2%	91.9%	99.8%	93.2%	91.2%	—	94.7%
		5	55	117	330	2	—	10	1	—	520
		1.0%	10.6%	22.5%	63.5%	0.4%	—	1.9%	0.2%	—	100.0%
		0.0%	0.1%	0.1%	0.8%	0.0%	—	0.0%	0.0%	—	0.1%
egrep	70	90.2%	94.4%	93.3%	91.0%	91.9%	99.8%	93.2%	91.2%	—	94.8%
		34	79	161	136	3	—	106	—	—	519
		6.6%	15.2%	31.0%	26.2%	0.6%	—	20.4%	—	—	100.0%
		0.1%	0.1%	0.2%	0.3%	0.0%	—	0.2%	—	—	0.1%
biff	71	90.3%	94.5%	93.5%	91.3%	91.9%	99.8%	93.4%	91.2%	—	94.9%
		43	90	107	140	80	—	37	11	—	508
		8.5%	17.7%	21.1%	27.6%	15.7%	—	7.3%	2.2%	—	100.0%
		0.1%	0.1%	0.1%	0.3%	0.2%	—	0.1%	0.1%	—	0.1%
rcp	72	90.5%	94.6%	93.6%	91.6%	92.2%	99.8%	93.5%	91.3%	—	95.0%
		28	126	166	58	84	—	24	19	—	505
		5.5%	25.0%	32.9%	11.5%	16.6%	—	4.8%	3.8%	—	100.0%
		0.1%	0.2%	0.2%	0.1%	0.2%	—	0.0%	0.2%	—	0.1%
df	73	90.5%	94.8%	93.8%	91.8%	92.4%	99.8%	93.6%	91.4%	—	95.1%
		29	8	132	29	4	—	18	280	—	500
		5.8%	1.6%	26.4%	5.8%	0.8%	—	3.6%	56.0%	—	100.0%
		0.1%	0.0%	0.1%	0.1%	0.0%	—	0.0%	2.4%	—	0.1%
chdir	74	90.6%	94.9%	93.9%	91.8%	92.4%	99.8%	93.6%	93.8%	—	95.2%
		39	120	275	37	4	—	24	—	—	499
		7.8%	24.0%	55.1%	7.4%	0.8%	—	4.8%	—	—	100.0%
		0.1%	0.2%	0.3%	0.1%	0.0%	—	0.0%	—	—	0.1%
case	75	90.8%	95.0%	94.2%	91.9%	92.4%	99.8%	93.6%	93.8%	—	95.3%
		31	104	118	103	81	—	59	1	—	497
		6.2%	20.9%	23.7%	20.7%	16.3%	—	11.9%	0.2%	—	100.0%
		0.1%	0.2%	0.1%	0.2%	0.2%	—	0.1%	0.0%	—	0.1%
bg	76	90.9%	95.2%	94.3%	92.2%	92.6%	99.8%	93.7%	93.8%	—	95.4%
		27	47	289	29	47	—	33	22	—	494
		5.5%	9.5%	58.5%	5.9%	9.5%	—	6.7%	4.5%	—	100.0%
		0.1%	0.1%	0.3%	0.1%	0.1%	—	0.1%	0.2%	—	0.1%
kill	77	90.9%	95.3%	94.6%	92.2%	92.8%	99.8%	93.8%	94.0%	—	95.5%
		1	—	66	230	124	—	12	36	—	469
		0.2%	—	14.1%	49.0%	26.4%	—	2.6%	7.7%	—	100.0%
		0.0%	—	0.1%	0.5%	0.3%	—	0.0%	0.3%	—	0.1%
Mail	78	90.9%	95.3%	94.7%	92.8%	93.1%	99.8%	93.8%	94.3%	—	95.6%
		1	230	83	73	15	—	59	—	—	461
		0.2%	49.9%	18.0%	15.8%	3.3%	—	12.8%	—	—	100.0%
		0.0%	0.4%	0.1%	0.2%	0.0%	—	0.1%	—	—	0.1%
breaksw	79	90.9%	95.7%	94.8%	93.0%	93.1%	99.8%	93.9%	94.3%	—	95.7%
		4	25	116	21	36	26	102	98	—	428
		0.9%	5.8%	27.1%	4.9%	8.4%	6.1%	23.8%	22.9%	—	100.0%
		0.0%	0.0%	0.1%	0.0%	0.1%	0.0%	0.2%	0.8%	—	0.1%
foreach	80	91.0%	95.7%	94.9%	93.0%	93.2%	99.8%	94.1%	95.2%	—	95.8%

Table 24 - One Hundred Commands from Each Class (6)

Command	Rank	Secretary	Facu	Grad	Guest	Misc-Human	Misc Non-Human	Prog	Root	Unknown	Totals
		14	34	68	105	47	—	115	7	—	390
		3.6%	8.7%	17.4%	26.9%	12.1%	—	29.5%	1.8%	—	100.0%
		0.0%	0.1%	0.1%	0.2%	0.1%	—	0.2%	0.1%	—	0.1%
w	81	91.0%	95.8%	94.9%	93.2%	93.4%	99.8%	94.3%	95.2%	—	95.9%
		—	1	57	107	28	—	161	—	—	354
		—	0.3%	16.1%	30.2%	7.9%	—	45.5%	—	—	100.0%
		—	0.0%	0.1%	0.3%	0.1%	—	0.3%	—	—	0.1%
sysline	82	91.0%	95.8%	95.0%	93.5%	93.4%	99.8%	94.6%	95.2%	—	96.0%
		17	125	143	9	25	—	28	6	—	353
		4.8%	35.4%	40.5%	2.5%	7.1%	—	7.9%	1.7%	—	100.0%
		0.1%	0.2%	0.1%	0.0%	0.1%	—	0.1%	0.1%	—	0.1%
history	83	91.1%	96.0%	95.1%	93.5%	93.5%	99.8%	94.7%	95.3%	—	96.0%
		—	—	4	169	126	—	53	—	—	352
		—	—	1.1%	48.0%	35.8%	—	15.1%	—	—	100.0%
		—	—	0.0%	0.4%	0.3%	—	0.1%	—	—	0.1%
rwhoson	84	91.1%	96.0%	95.1%	93.9%	93.9%	99.8%	94.8%	95.3%	—	96.1%
		65	24	78	52	56	—	58	2	—	335
		19.4%	7.2%	23.3%	15.5%	16.7%	—	17.3%	0.6%	—	100.0%
		0.2%	0.0%	0.1%	0.1%	0.2%	—	0.1%	0.0%	—	0.1%
talk	85	91.3%	96.0%	95.2%	94.0%	94.0%	99.8%	94.9%	95.3%	—	96.2%
		—	12	57	—	231	—	32	—	—	332
		—	3.6%	17.2%	—	69.6%	—	9.6%	—	—	100.0%
		—	0.0%	0.1%	—	0.6%	—	0.1%	—	—	0.1%
obj	86	91.3%	96.0%	95.3%	94.0%	94.6%	99.8%	94.9%	95.3%	—	96.2%
		322	1	1	—	1	—	—	—	—	325
		99.1%	0.3%	0.3%	—	0.3%	—	—	—	—	100.0%
		1.0%	0.0%	0.0%	—	0.0%	—	—	—	—	0.1%
mc	87	92.3%	96.0%	95.3%	94.0%	94.6%	99.8%	94.9%	95.3%	—	96.3%
		—	247	43	7	—	—	26	—	—	323
		—	76.5%	13.3%	2.2%	—	—	8.0%	—	—	100.0%
		—	0.4%	0.0%	0.0%	—	—	0.0%	—	—	0.1%
sccs	88	92.3%	96.4%	95.3%	94.1%	94.6%	99.8%	95.0%	95.3%	—	96.4%
		—	1	1	314	—	—	—	—	—	316
		—	0.3%	0.3%	99.4%	—	—	—	—	—	100.0%
		—	0.0%	0.0%	0.7%	—	—	—	—	—	0.1%
0	89	92.3%	96.4%	95.3%	94.8%	94.6%	99.8%	95.0%	95.3%	—	96.4%
		152	35	67	23	1	—	11	—	—	289
		52.6%	12.1%	23.2%	8.0%	0.3%	—	3.8%	—	—	100.0%
		0.5%	0.1%	0.1%	0.1%	0.0%	—	0.0%	—	—	0.1%
cc	90	92.8%	96.5%	95.4%	94.8%	94.6%	99.8%	95.0%	95.3%	—	96.5%
		7	44	152	13	41	—	22	2	—	281
		2.5%	15.7%	54.1%	4.6%	14.6%	—	7.8%	0.7%	—	100.0%
		0.0%	0.1%	0.2%	0.0%	0.1%	—	0.0%	0.0%	—	0.1%
mkdir	91	92.8%	96.5%	95.5%	94.9%	94.8%	99.8%	95.1%	95.3%	—	96.6%
		22	90	48	54	34	—	10	1	—	259
		8.5%	34.7%	18.5%	20.8%	13.1%	—	3.9%	0.4%	—	100.0%
		0.1%	0.1%	0.0%	0.1%	0.1%	—	0.0%	0.0%	—	0.1%
tail	92	92.8%	96.7%	95.6%	95.0%	94.8%	99.8%	95.1%	95.3%	—	96.6%
		11	32	61	22	30	39	45	17	—	257
		4.3%	12.5%	23.7%	8.6%	11.7%	15.2%	17.5%	6.6%	—	100.0%
		0.0%	0.1%	0.1%	0.1%	0.1%	0.0%	0.1%	0.1%	—	0.1%
chmod	93	92.9%	96.7%	95.7%	95.1%	94.9%	99.8%	95.2%	95.5%	—	96.7%
		1	26	156	17	30	—	22	3	—	255
		0.4%	10.2%	61.2%	6.7%	11.8%	—	8.6%	1.2%	—	100.0%
		0.0%	0.0%	0.2%	0.0%	0.1%	—	0.0%	0.0%	—	0.1%
fgrep	94	92.9%	96.8%	95.8%	95.1%	95.0%	99.8%	95.2%	95.5%	—	96.7%
		8	5	11	17	214	—	—	—	—	255
		3.1%	2.0%	4.3%	6.7%	83.9%	—	—	—	—	100.0%
		0.0%	0.0%	0.0%	0.0%	0.6%	—	—	—	—	0.1%
ln	95	92.9%	96.8%	95.8%	95.1%	95.6%	99.8%	95.2%	95.5%	—	96.8%
		115	62	36	—	1	—	34	—	—	248
		46.4%	25.0%	14.5%	—	0.4%	—	13.7%	—	—	100.0%
		0.4%	0.1%	0.0%	—	0.0%	—	0.1%	—	—	0.1%
troff	96	93.3%	96.9%	95.9%	95.1%	95.6%	99.8%	95.3%	95.5%	—	96.8%

Table 24 - One Hundred Commands from Each Class (7)

Command	Rank	Secretary	Facu	Grad	Guest	Misc-Human	Misc Non-Human	Prog	Root	Unknown	Totals
		9	—	78	12	3	—	141	—	—	243
		3.7%	—	32.1%	4.9%	1.2%	—	58.0%	—	—	100.0%
		0.0%	—	0.1%	0.0%	0.0%	—	0.3%	—	—	0.1%
mesg	97	93.3%	96.9%	95.9%	95.2%	95.6%	99.8%	95.5%	95.5%	—	96.9%
		73	—	16	119	19	—	—	—	—	227
		32.2%	—	7.0%	52.4%	8.4%	—	—	—	—	100.0%
		0.2%	—	0.0%	0.3%	0.1%	—	—	—	—	0.0%
page	98	93.5%	96.9%	96.0%	95.4%	95.7%	99.8%	95.5%	95.5%	—	96.9%
		12	—	186	5	8	—	9	4	—	224
		5.4%	—	83.0%	2.2%	3.6%	—	4.0%	1.8%	—	100.0%
		0.0%	—	0.2%	0.0%	0.0%	—	0.0%	0.0%	—	0.0%
tar	99	93.6%	96.9%	96.1%	95.5%	95.7%	99.8%	95.5%	95.5%	—	97.0%
		105	50	32	—	1	—	34	—	—	222
		47.3%	22.5%	14.4%	—	0.5%	—	15.3%	—	—	100.0%
		0.3%	0.1%	0.0%	—	0.0%	—	0.1%	—	—	0.0%
itroff	100	93.9%	97.0%	96.2%	95.5%	95.7%	99.8%	95.6%	95.5%	—	97.0%
		31711	62317	99220	42583	36623	140294	54117	11671	—	478536
		6.6%	13.0%	20.7%	8.9%	7.7%	29.3%	11.3%	2.4%	—	100.0%
		100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Totals	—	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

4.2.5. Alias and Background Details for Specific Commands

Table 25, "Command Totals for 500 Commands," shows the relative use of the 500 most frequently used commands. How often each command appears in a command line using alias substitution and how often each command appears in a command line executed in the background is also shown. The "Overall Frequency" column shows how many times the command was executed. The "Per Cent of All Commands" column shows the ratio of the entry in the previous column to the total number of command executions. The "Cumulative Per Cent" column shows the cumulative totals of the previous column. The "Alias Per Cent" column shows the ratio of the number of times the command appeared in a command line using alias substitutions, to the total number of times the command was executed. The "Background Per Cent" column shows the ratio of the number of times the command appeared in a command line that began execution in the background to the total number of times the command was executed. While the analysis does not show *which* command in a command line using alias substitution actually was aliased, one can infer that a high "alias per cent" implies that the command was often aliased, since the vast majority of all command lines only contain one command. Note that *ls* was aliased over three-fourths of the time. Users often alias *l* to *ls*, or alias *ls options* to *ls*, where *options* are options that the user prefers over the default *ls* options. Other commands that are aliased over half of the time they are used include *jobs*, *rwho*, *pushd*, *egrep*, *chdir*, *Mail*, *history*, and *fgrep*. Commands that are executed in the background over 20 per cent of the time include *make*, *ditroff*, *rcp*, and *itroff*, all commands which take a relatively long time to execute.

Table 26, "Five Hundred Command Totals from Terminal," gives the same information as Table 25, except it includes only data from commands typed by users at their terminals. Note that 14 commands account for half of all commands executed by users at their terminals.

Table 25 - Command Totals for 500 Commands (1)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
alias	1	106805	22.3%	22.3%	33.1%	—
set	2	62342	13.0%	35.3%	1.4%	—
if	3	48085	10.0%	45.4%	0.6%	—
shift	4	24089	5.0%	50.4%	—	—
switch	5	19009	4.0%	54.4%	—	—
goto	6	18115	3.8%	58.2%	—	—
echo	7	14882	3.1%	61.3%	14.2%	0.3%
setenv	8	13481	2.8%	64.1%	0.5%	—
ls	9	11227	2.3%	66.5%	77.0%	0.1%
endif	10	10864	2.3%	68.7%	—	—
exit	11	7337	1.5%	70.3%	2.0%	—
end	12	6858	1.4%	71.7%	—	—
rm	13	5731	1.2%	72.9%	17.4%	0.1%
cd	14	5638	1.2%	74.1%	10.3%	—
umask	15-	5372	1.1%	75.2%	0.1%	—
vi	16	5280	1.1%	76.3%	0.5%	—
cat	17	5102	1.1%	77.4%	6.6%	0.1%
more	18	4389	0.9%	78.3%	46.0%	0.1%
LABEL	19	4237	0.9%	79.2%	—	—
jobs	20	4009	0.8%	80.0%	69.2%	—
unset	21	3683	0.8%	80.8%	5.9%	—
while	22	3418	0.7%	81.5%	—	—
stty	23	3247	0.7%	82.2%	3.1%	—
@	24	3082	0.6%	82.8%	0.0%	—
fg	25	2888	0.6%	83.4%	0.5%	0.0%
ips	26	2813	0.6%	84.0%	—	—
onintr	27	2651	0.6%	84.6%	—	—
eval	28	2362	0.5%	85.1%	17.2%	—
mail	29	2353	0.5%	85.5%	11.3%	3.3%
source	30	2346	0.5%	86.0%	18.2%	—
clear	31	2265	0.5%	86.5%	28.7%	—
msgs	32	2180	0.5%	87.0%	20.5%	0.4%
sleep	33	1870	0.4%	87.4%	2.5%	2.6%
else	34	1837	0.4%	87.7%	—	—
ps	35	1741	0.4%	88.1%	4.9%	0.8%
mv	36	1694	0.4%	88.5%	31.2%	0.1%
dit-imagen	37	1483	0.3%	88.8%	—	—
rlogin	38	1392	0.3%	89.1%	18.5%	0.3%
la	39	1355	0.3%	89.3%	—	—
date	40	1279	0.3%	89.6%	1.7%	0.1%

Table 25 - Command Totals for 500 Commands (2)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
logout	41	1223	0.3%	89.9%	35.9%	—
u	42	1172	0.2%	90.1%	—	—
ruptime	43	1163	0.2%	90.3%	43.3%	—
pwd	44	1157	0.2%	90.6%	18.3%	—
finger	45	1148	0.2%	90.8%	35.3%	0.3%
fortune	46	1105	0.2%	91.1%	5.5%	—
grep	47	1009	0.2%	91.3%	36.4%	0.2%
4	48	996	0.2%	91.5%	—	—
catdvi	49	963	0.2%	91.7%	—	—
rsh	50	953	0.2%	91.9%	28.9%	6.7%
dviimp	51	948	0.2%	92.1%	—	—
%	52	930	0.2%	92.3%	2.0%	0.3%
lpq	53	907	0.2%	92.5%	42.8%	1.9%
dirs	54	834	0.2%	92.6%	29.0%	—
f	55	827	0.2%	92.8%	—	0.1%
ipq	56	761	0.2%	93.0%	—	—
uptime	57	706	0.1%	93.1%	19.3%	—
from	58	695	0.1%	93.3%	—	—
who	59	663	0.1%	93.4%	5.3%	—
lpr	60	654	0.1%	93.5%	13.1%	10.4%
make	61	634	0.1%	93.7%	27.0%	29.3%
$troff	62	632	0.1%	93.8%	—	—
rwho	63	618	0.1%	93.9%	61.0%	0.3%
$lpr	64	609	0.1%	94.1%	—	—
cp	65	605	0.1%	94.2%	30.4%	0.5%
pushd	66	603	0.1%	94.3%	71.1%	—
pi	67	602	0.1%	94.4%	12.0%	14.6%
man	68	576	0.1%	94.6%	1.9%	0.2%
ditroff	69	544	0.1%	94.7%	22.1%	58.8%
egrep	70	520	0.1%	94.8%	55.0%	1.0%
biff	71	519	0.1%	94.9%	0.2%	—
rcp	72	508	0.1%	95.0%	16.5%	21.7%
df	73	505	0.1%	95.1%	0.8%	—
chdir	74	500	0.1%	95.2%	99.8%	—
case	75	499	0.1%	95.3%	—	—
bg	76	497	0.1%	95.4%	0.6%	—
kill	77	494	0.1%	95.5%	43.7%	—
Mail	78	469	0.1%	95.6%	55.4%	0.2%
breaksw	79	461	0.1%	95.7%	—	—
foreach	80	428	0.1%	95.8%	—	—

Table 25 - Command Totals for 500 Commands (3)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
w	81	390	0.1%	95.9%	4.9%	0.3%
sysline	82	354	0.1%	96.0%	—	0.3%
history	83	353	0.1%	96.0%	89.5%	—
rwhoson	84	352	0.1%	96.1%	29.8%	—
talk	85	335	0.1%	96.2%	2.4%	0.9%
obj	86	332	0.1%	96.2%	0.3%	1.8%
mc	87	325	0.1%	96.3%	—	—
sccs	88	323	0.1%	96.4%	0.6%	—
0	89	316	0.1%	96.4%	—	—
cc	90	289	0.1%	96.5%	8.7%	9.7%
mkdir	91	281	0.1%	96.6%	1.8%	—
tail	92	259	0.1%	96.6%	20.5%	—
chmod	93	257	0.1%	96.7%	7.0%	3.1%
fgrep	94	255	0.1%	96.7%	69.0%	0.8%
ln	95	255	0.1%	96.8%	0.4%	0.4%
troff	96	248	0.1%	96.8%	—	—
mesg	97	243	0.1%	96.9%	—	—
page	98	227	0.0%	96.9%	6.2%	—
tar	99	224	0.0%	97.0%	0.9%	12.9%
itroff	100	222	0.0%	97.0%	41.9%	81.1%
popd	101	220	0.0%	97.1%	69.5%	—
robots	102	215	0.0%	97.1%	—	—
last	103	200	0.0%	97.2%	—	—
tty	104	198	0.0%	97.2%	—	—
endsw	105	196	0.0%	97.2%	—	—
sendmail	106	184	0.0%	97.3%	3.3%	4.3%
a.out	107	173	0.0%	97.3%	—	2.9%
dl	108	166	0.0%	97.3%	97.6%	—
tset	109	166	0.0%	97.4%	5.4%	—
show	110	165	0.0%	97.4%	—	—
su	111	165	0.0%	97.4%	—	—
q	112	164	0.0%	97.5%	—	—
du	113	163	0.0%	97.5%	0.6%	—
inc	114	159	0.0%	97.5%	—	—
idle	115	157	0.0%	97.6%	—	—
wc	116	150	0.0%	97.6%	32.0%	—
head	117	149	0.0%	97.6%	32.2%	—
nroff	118	146	0.0%	97.7%	—	6.2%
n	119	144	0.0%	97.7%	—	—
dpq	120	143	0.0%	97.7%	—	—

Table 25 - Command Totals for 500 Commands (4)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
telnet	121	143	0.0%	97.8%	10.5%	—
csh	122	142	0.0%	97.8%	—	4.2%
yank	123	132	0.0%	97.8%	—	—
limit	124	131	0.0%	97.8%	—	—
luk	125	125	0.0%	97.9%	—	—
find	126	124	0.0%	97.9%	—	4.8%
px	127	120	0.0%	97.9%	—	—
sort	128	119	0.0%	98.0%	72.3%	0.8%
rogue	129	118	0.0%	98.0%	36.4%	—
l	130	117	0.0%	98.0%	0.9%	—
dvi-imagen	131	115	0.0%	98.0%	—	—
next	132	113	0.0%	98.0%	—	—
suspend	133	112	0.0%	98.1%	93.8%	—
tbl	134	109	0.0%	98.1%	—	80.7%
whom	135	103	0.0%	98.1%	2.9%	—
go	136	101	0.0%	98.1%	—	—
ftp	137	100	0.0%	98.2%	—	—
time	138	97	0.0%	98.2%	—	—
retrieve	139	96	0.0%	98.2%	72.9%	—
rmm	140	95	0.0%	98.2%	—	—
diff	141	93	0.0%	98.2%	2.2%	4.3%
nice	142	90	0.0%	98.3%	27.8%	15.6%
view	143	88	0.0%	98.3%	—	—
pdx	144	87	0.0%	98.3%	—	—
rmdir	145	85	0.0%	98.3%	—	—
file	146	77	0.0%	98.3%	—	—
pic	147	77	0.0%	98.3%	—	10.4%
uuclean	148	77	0.0%	98.4%	—	—
whoami	149	77	0.0%	98.4%	—	—
break	150	75	0.0%	98.4%	—	—
scan	151	73	0.0%	98.4%	—	—
ucbvax	152	73	0.0%	98.4%	5.5%	17.8%
pr	153	72	0.0%	98.4%	34.7%	4.2%
ping	154	70	0.0%	98.4%	—	—
NULL	155	69	0.0%	98.5%	—	—
write	156	69	0.0%	98.5%	26.1%	—
vpq	157	67	0.0%	98.5%	73.1%	—
sync	158	66	0.0%	98.5%	—	—
uc	159	65	0.0%	98.5%	—	—
fing	160	61	0.0%	98.5%	—	—

Table 25 - Command Totals for 500 Commands (5)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
rrn	161	60	0.0%	98.5%	—	—
unalias	162	60	0.0%	98.6%	10.0%	—
checknr	163	59	0.0%	98.6%	91.5%	—
deqn	164	59	0.0%	98.6%	1.7%	40.7%
look	165	59	0.0%	98.6%	16.9%	—
printenv	166	59	0.0%	98.6%	28.8%	—
whereis	167	59	0.0%	98.6%	—	—
j	168	58	0.0%	98.6%	—	—
netstat	169	58	0.0%	98.6%	1.7%	1.7%
rsl	170	56	0.0%	98.7%	28.6%	5.4%
rdist	171	55	0.0%	98.7%	14.5%	12.7%
test	172	55	0.0%	98.7%	1.8%	—
r	173	54	0.0%	98.7%	—	—
users	174	54	0.0%	98.7%	—	—
vfontedpr	175	54	0.0%	98.7%	—	—
client	176	53	0.0%	98.7%	79.2%	—
Spell	177	52	0.0%	98.7%	—	—
radx	178	52	0.0%	98.7%	3.8%	5.8%
at	179	51	0.0%	98.8%	—	—
glob	180	51	0.0%	98.8%	7.8%	—
*	181	49	0.0%	98.8%	2.0%	—
cas	182	49	0.0%	98.8%	—	—
hm	183	48	0.0%	98.8%	—	—
i	184	48	0.0%	98.8%	—	—
quiz	185	48	0.0%	98.8%	—	—
awk	186	47	0.0%	98.8%	—	—
ss	187	47	0.0%	98.8%	—	—
uniq	188	47	0.0%	98.8%	63.8%	—
get	189	46	0.0%	98.9%	—	—
getNAME	190	45	0.0%	98.9%	71.1%	—
hostname	191	45	0.0%	98.9%	—	—
on	192	44	0.0%	98.9%	—	2.3%
soc	193	44	0.0%	98.9%	52.3%	4.5%
spell	194	41	0.0%	98.9%	—	26.8%
2	195	40	0.0%	98.9%	—	—
lastcomm	196	40	0.0%	98.9%	5.0%	—
t	197	40	0.0%	98.9%	—	—
rehash	198	39	0.0%	98.9%	12.8%	—
mt	199	38	0.0%	98.9%	—	31.6%
strings	200	38	0.0%	99.0%	—	—

Table 25 - Command Totals for 500 Commands (6)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
dump	201	37	0.0%	99.0%	—	—
mlog	202	37	0.0%	99.0%	—	—
}	203	37	0.0%	99.0%	—	—
chfn	204	36	0.0%	99.0%	—	—
dbx	205	35	0.0%	99.0%	—	—
ia	206	35	0.0%	99.0%	—	—
lll-crg	207	35	0.0%	99.0%	—	5.7%
old_copy2.	208	34	0.0%	99.0%	—	—
continue	209	33	0.0%	99.0%	—	—
whois	210	33	0.0%	99.0%	—	—
ex	211	32	0.0%	99.0%	—	—
tee	212	32	0.0%	99.0%	6.3%	—
dd	213	31	0.0%	99.0%	19.4%	83.9%
repl	214	31	0.0%	99.1%	—	—
print.xero	215	30	0.0%	99.1%	—	—
'lpr'	216	29	0.0%	99.1%	100.0%	100.0%
chown	217	29	0.0%	99.1%	—	—
grn	218	29	0.0%	99.1%	—	34.5%
lprm	219	29	0.0%	99.1%	6.9%	—
m	220	28	0.0%	99.1%	—	—
nm	221	28	0.0%	99.1%	—	—
pc	222	28	0.0%	99.1%	10.7%	14.3%
s	223	27	0.0%	99.1%	—	—
OK	224	26	0.0%	99.1%	—	—
makewhatis	225	26	0.0%	99.1%	—	3.8%
mboxsize	226	26	0.0%	99.1%	—	—
newesl	227	26	0.0%	99.1%	—	—
sh	228	26	0.0%	99.1%	—	30.8%
yacc	229	26	0.0%	99.1%	—	34.6%
eqn	230	25	0.0%	99.1%	12.0%	72.0%
initex	231	25	0.0%	99.1%	—	—
lpc	232	24	0.0%	99.2%	—	—
run	233	24	0.0%	99.2%	8.3%	79.2%
sed	234	24	0.0%	99.2%	—	25.0%
vgrind	235	24	0.0%	99.2%	4.2%	41.7%
cmp	236	23	0.0%	99.2%	—	—
e	237	23	0.0%	99.2%	65.2%	—
gprof	238	23	0.0%	99.2%	—	26.1%
pw	239	23	0.0%	99.2%	—	—
rn	240	23	0.0%	99.2%	—	—

Table 25 - Command Totals for 500 Commands (7)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
apropos	241	22	0.0%	99.2%	—	—
refer	242	22	0.0%	99.2%	4.5%	27.3%
vtroff	243	22	0.0%	99.2%	—	4.5%
y	244	22	0.0%	99.2%	—	—
"."	245	21	0.0%	99.2%	—	—
dc	246	21	0.0%	99.2%	—	—
dtbl	247	21	0.0%	99.2%	4.8%	71.4%
eslrex	248	21	0.0%	99.2%	—	—
magic	249	21	0.0%	99.2%	—	—
pwho	250	21	0.0%	99.2%	100.0%	—
{	251	21	0.0%	99.2%	—	—
fix_me	252	20	0.0%	99.2%	—	—
mine	253	20	0.0%	99.2%	—	—
modsite	254	20	0.0%	99.3%	5.0%	—
prmail	255	20	0.0%	99.3%	—	—
script	256	20	0.0%	99.3%	—	—
systat	257	20	0.0%	99.3%	—	—
cal	258	19	0.0%	99.3%	—	—
ed	259	19	0.0%	99.3%	—	—
expn	260	19	0.0%	99.3%	5.3%	—
netnews	261	19	0.0%	99.3%	—	—
reset	262	19	0.0%	99.3%	5.3%	—
restore	263	19	0.0%	99.3%	—	—
tn3270	264	19	0.0%	99.3%	—	—
touch	265	19	0.0%	99.3%	36.8%	—
vial	266	19	0.0%	99.3%	—	—
`?	267	19	0.0%	99.3%	—	—
$sort	268	18	0.0%	99.3%	—	—
bye	269	18	0.0%	99.3%	—	—
iostat	270	18	0.0%	99.3%	—	—
pathalias	271	18	0.0%	99.3%	—	—
rup	272	18	0.0%	99.3%	—	—
!	273	17	0.0%	99.3%	—	5.9%
ap.out	274	17	0.0%	99.3%	—	—
emacs	275	17	0.0%	99.3%	—	—
b	276	16	0.0%	99.3%	—	87.5%
d	277	16	0.0%	99.3%	—	—
mpx	278	16	0.0%	99.3%	—	—
mysh	279	16	0.0%	99.3%	—	—
ERROR	280	15	0.0%	99.4%	—	—

Table 25 - Command Totals for 500 Commands (8)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
compress	281	15	0.0%	99.4%	53.3%	26.7%
i.out	282	15	0.0%	99.4%	—	—
initialize	283	15	0.0%	99.4%	46.7%	—
mille	284	15	0.0%	99.4%	26.7%	—
pd	285	15	0.0%	99.4%	—	—
rwhom	286	15	0.0%	99.4%	53.3%	—
uu	287	15	0.0%	99.4%	—	—
vipw	288	15	0.0%	99.4%	—	—
colrm	289	14	0.0%	99.4%	14.3%	—
h	290	14	0.0%	99.4%	—	—
me	291	14	0.0%	99.4%	—	—
nc	292	14	0.0%	99.4%	—	—
p	293	14	0.0%	99.4%	—	—
rep	294	14	0.0%	99.4%	—	—
whoson	295	14	0.0%	99.4%	—	—
comp	296	13	0.0%	99.4%	—	—
getdu	297	13	0.0%	99.4%	—	—
csrg	298	12	0.0%	99.4%	100.0%	—
extern	299	12	0.0%	99.4%	—	—
jitter	300	12	0.0%	99.4%	—	—
leave	301	12	0.0%	99.4%	—	—
nfontcheck	302	12	0.0%	99.4%	58.3%	—
tex	303	12	0.0%	99.4%	—	—
tool	304	12	0.0%	99.4%	—	—
undump	305	12	0.0%	99.4%	—	—
unsetenv	306	12	0.0%	99.4%	16.7%	—
window	307	12	0.0%	99.4%	—	—
x	308	12	0.0%	99.4%	—	—
LS	309	11	0.0%	99.4%	—	—
c	310	11	0.0%	99.4%	—	—
ccom	311	11	0.0%	99.4%	—	—
findit	312	11	0.0%	99.4%	—	—
foo	313	11	0.0%	99.4%	—	100.0%
groups	314	11	0.0%	99.4%	—	—
ll	315	11	0.0%	99.4%	—	—
owd	316	11	0.0%	99.4%	—	—
pix	317	11	0.0%	99.5%	—	—
rslcreate	318	11	0.0%	99.5%	36.4%	—
split	319	11	0.0%	99.5%	—	9.1%
timedc	320	11	0.0%	99.5%	—	—

Table 25 - Command Totals for 500 Commands (9)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
uncompress	321	11	0.0%	99.5%	81.8%	9.1%
whatis	322	11	0.0%	99.5%	—	—
]	323	10	0.0%	99.5%	—	—
a	324	10	0.0%	99.5%	20.0%	—
dpcheck	325	10	0.0%	99.5%	—	—
exec	326	10	0.0%	99.5%	—	100.0%
palias	327	10	0.0%	99.5%	20.0%	40.0%
passwd	328	10	0.0%	99.5%	—	—
prev	329	10	0.0%	99.5%	—	—
renice	330	10	0.0%	99.5%	—	—
which	331	10	0.0%	99.5%	—	10.0%
`]	332	9	0.0%	99.5%	—	—
'lpq'	333	9	0.0%	99.5%	88.9%	88.9%
adb	334	9	0.0%	99.5%	—	—
bib	335	9	0.0%	99.5%	—	22.2%
dpstart	336	9	0.0%	99.5%	—	—
equel	337	9	0.0%	99.5%	—	—
hello	338	9	0.0%	99.5%	—	—
mo	339	9	0.0%	99.5%	—	—
rdump	340	9	0.0%	99.5%	—	—
runss	341	9	0.0%	99.5%	—	55.6%
scalepx	342	9	0.0%	99.5%	77.8%	44.4%
sp-	343	9	0.0%	99.5%	—	—
tr	344	9	0.0%	99.5%	—	—
vcp	345	9	0.0%	99.5%	—	—
Z	346	8	0.0%	99.5%	—	—
ar	347	8	0.0%	99.5%	—	—
atq	348	8	0.0%	99.5%	—	—
co	349	8	0.0%	99.5%	—	—
didrs	350	8	0.0%	99.5%	—	—
fontcheck	351	8	0.0%	99.5%	25.0%	—
gnumacs	352	8	0.0%	99.5%	—	—
igrind	353	8	0.0%	99.5%	—	75.0%
mod2.0$v	354	8	0.0%	99.5%	—	—
sl	355	8	0.0%	99.5%	—	—
ul	356	8	0.0%	99.5%	75.0%	—
uucp	357	8	0.0%	99.5%	—	—
wotd	358	8	0.0%	99.5%	—	—
&	359	7	0.0%	99.5%	—	85.7%
".PP"	360	7	0.0%	99.5%	—	—

Table 25 - Command Totals for 500 Commands (10)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
".sp"	361	7	0.0%	99.5%	—	—
atrm	362	7	0.0%	99.5%	—	—
dterm	363	7	0.0%	99.5%	—	—
f77	364	7	0.0%	99.5%	—	100.0%
gn	365	7	0.0%	99.5%	—	—
gomoku	366	7	0.0%	99.5%	—	—
inp.o	367	7	0.0%	99.5%	—	14.3%
ireqn	368	7	0.0%	99.5%	71.4%	85.7%
job	369	7	0.0%	99.5%	14.3%	—
jos	370	7	0.0%	99.5%	—	—
k	371	7	0.0%	99.5%	—	—
lks	372	7	0.0%	99.5%	—	—
mai	373	7	0.0%	99.6%	28.6%	—
mroe	374	7	0.0%	99.6%	—	—
named	375	7	0.0%	99.6%	—	—
opq	376	7	0.0%	99.6%	—	—
printlabs	377	7	0.0%	99.6%	—	—
rstat	378	7	0.0%	99.6%	—	—
ru	379	7	0.0%	99.6%	—	—
sentinfo	380	7	0.0%	99.6%	—	—
sr	381	7	0.0%	99.6%	—	—
tip	382	7	0.0%	99.6%	—	—
uuencode	383	7	0.0%	99.6%	—	57.1%
vmstat	384	7	0.0%	99.6%	—	—
^X	385	6	0.0%	99.6%	—	—
^]z	386	6	0.0%	99.6%	—	—
,	387	6	0.0%	99.6%	—	—
1	388	6	0.0%	99.6%	—	—
chgrp	389	6	0.0%	99.6%	—	—
ci	390	6	0.0%	99.6%	—	—
delayben.c	391	6	0.0%	99.6%	—	83.3%
digrind	392	6	0.0%	99.6%	66.7%	100.0%
fish	393	6	0.0%	99.6%	—	—
fix_stats	394	6	0.0%	99.6%	—	—
hashstat	395	6	0.0%	99.6%	—	—
implot	396	6	0.0%	99.6%	—	—
jbos	397	6	0.0%	99.6%	—	—
lock	398	6	0.0%	99.6%	—	—
login	399	6	0.0%	99.6%	—	—
logut	400	6	0.0%	99.6%	—	—

Table 25 - Command Totals for 500 Commands (11)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
lsf	401	6	0.0%	99.6%	—	—
parentg	402	6	0.0%	99.6%	—	—
watchhost	403	6	0.0%	99.6%	—	—
xp	404	6	0.0%	99.6%	—	—
z	405	6	0.0%	99.6%	—	—
^H	406	5	0.0%	99.6%	—	—
$i	407	5	0.0%	99.6%	—	—
From	408	5	0.0%	99.6%	—	—
addcal	409	5	0.0%	99.6%	—	—
adventure	410	5	0.0%	99.6%	—	—
call	411	5	0.0%	99.6%	—	—
ccd	412	5	0.0%	99.6%	—	—
checknews	413	5	0.0%	99.6%	—	—
chsh	414	5	0.0%	99.6%	—	—
cle	415	5	0.0%	99.6%	—	—
comm	416	5	0.0%	99.6%	—	—
corbomite	417	5	0.0%	99.6%	—	—
folder	418	5	0.0%	99.6%	—	—
getpaper	419	5	0.0%	99.6%	—	—
help	420	5	0.0%	99.6%	—	—
idi	421	5	0.0%	99.6%	100.0%	100.0%
iq	422	5	0.0%	99.6%	—	—
jj	423	5	0.0%	99.6%	—	—
ld	424	5	0.0%	99.6%	—	—
lo	425	5	0.0%	99.6%	—	—
lss	426	5	0.0%	99.6%	—	—
m.out	427	5	0.0%	99.6%	—	—
notify	428	5	0.0%	99.6%	80.0%	—
oparentg	429	5	0.0%	99.6%	—	—
postnews	430	5	0.0%	99.6%	—	—
printl	431	5	0.0%	99.6%	—	20.0%
probabilit	432	5	0.0%	99.6%	—	—
pwdd	433	5	0.0%	99.6%	—	—
smtp	434	5	0.0%	99.6%	—	—
sortm	435	5	0.0%	99.6%	—	40.0%
stat	436	5	0.0%	99.6%	—	—
strip	437	5	0.0%	99.6%	—	—
tree	438	5	0.0%	99.6%	—	—
v	439	5	0.0%	99.6%	—	—
virtex	440	5	0.0%	99.6%	—	—

Table 25 - Command Totals for 500 Commands (12)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
webster	441	5	0.0%	99.6%	20.0%	—
wpd	442	5	0.0%	99.6%	—	—
zmore	443	5	0.0%	99.6%	—	—
^L	444	4	0.0%	99.6%	—	—
"rm"	445	4	0.0%	99.6%	—	—
#include	446	4	0.0%	99.6%	—	—
".ds"	447	4	0.0%	99.6%	—	—
10	448	4	0.0%	99.6%	—	—
V.2.2	449	4	0.0%	99.6%	—	—
[	450	4	0.0%	99.6%	—	—
addbib	451	4	0.0%	99.6%	—	—
al	452	4	0.0%	99.6%	—	—
ca	453	4	0.0%	99.6%	—	—
ctags	454	4	0.0%	99.6%	—	—
daemon	455	4	0.0%	99.6%	—	25.0%
ddirs	456	4	0.0%	99.6%	—	—
dgr	457	4	0.0%	99.6%	—	—
dgs	458	4	0.0%	99.6%	—	—
fetch	459	4	0.0%	99.6%	—	—
figner	460	4	0.0%	99.6%	—	—
fmt	461	4	0.0%	99.6%	—	—
fn	462	4	0.0%	99.6%	—	—
forw	463	4	0.0%	99.6%	—	—
gre	464	4	0.0%	99.7%	—	—
int	465	4	0.0%	99.7%	—	—
ip	466	4	0.0%	99.7%	—	—
kj	467	4	0.0%	99.7%	—	—
lbl-csam	468	4	0.0%	99.7%	—	—
ll.bib	469	4	0.0%	99.7%	—	—
lpiq	470	4	0.0%	99.7%	—	—
mal	471	4	0.0%	99.7%	—	—
mbox	472	4	0.0%	99.7%	—	—
mod2.1$v	473	4	0.0%	99.7%	—	—
mount	474	4	0.0%	99.7%	—	—
nray	475	4	0.0%	99.7%	—	100.0%
nta-vax	476	4	0.0%	99.7%	—	—
out	477	4	0.0%	99.7%	50.0%	25.0%
pargv	478	4	0.0%	99.7%	—	—
printi	479	4	0.0%	99.7%	—	100.0%
prompt.wul	480	4	0.0%	99.7%	—	—

Table 25 - Command Totals for 500 Commands						(13)
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
pwdu	481	4	0.0%	99.7%	—	—
query	482	4	0.0%	99.7%	—	—
repeat	483	4	0.0%	99.7%	—	25.0%
rmf	484	4	0.0%	99.7%	—	—
robtest	485	4	0.0%	99.7%	—	—
rsl2	486	4	0.0%	99.7%	—	100.0%
ruptie	487	4	0.0%	99.7%	—	—
rwhoso	488	4	0.0%	99.7%	—	—
smgs	489	4	0.0%	99.7%	—	—
stop	490	4	0.0%	99.7%	25.0%	—
uncompact	491	4	0.0%	99.7%	—	—
wait	492	4	0.0%	99.7%	—	—
wd	493	4	0.0%	99.7%	—	—
wump	494	4	0.0%	99.7%	—	—
zork	495	4	0.0%	99.7%	—	—
!-	496	3	0.0%	99.7%	—	—
"	497	3	0.0%	99.7%	—	—
".nr"	498	3	0.0%	99.7%	—	—
411	499	3	0.0%	99.7%	—	—
6	500	3	0.0%	99.7%	—	—
Total	—	478536	100.0%	100.0%	13.7%	0.4%

Table 26 - Five Hundred Command Totals from Terminal (1)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
ls	1	11224	12.4%	12.4%	9.5%	0.0%
vi	2	5280	5.8%	18.2%	0.0%	0.0%
cd	3	5158	5.7%	23.9%	0.6%	0.0%
more	4	4017	4.4%	28.3%	2.1%	0.0%
jobs	5	4005	4.4%	32.7%	3.1%	0.0%
fg	6	2888	3.2%	35.9%	0.0%	0.0%
echo	7	2817	3.1%	39.0%	2.3%	0.1%
mail	8	2329	2.6%	41.6%	0.3%	0.1%
ps	9	1739	1.9%	43.5%	0.1%	0.0%
rm	10	1723	1.9%	45.4%	1.1%	0.0%
la	11	1276	1.4%	46.8%	0.0%	0.0%
rlogin	12	1267	1.4%	48.2%	0.3%	0.0%
logout	13	1222	1.3%	49.5%	0.5%	0.0%
cat	14	1174	1.3%	50.8%	0.3%	0.0%
u	15	1164	1.3%-	52.1%	0.0%	0.0%
finger	16	1148	1.3%	53.4%	0.4%	0.0%
exit	17	1005	1.1%	54.5%	0.2%	0.0%
4	18	996	1.1%	55.6%	0.0%	0.0%
grep	19	979	1.1%	56.6%	0.4%	0.0%
pwd	20	953	1.0%	57.7%	0.2%	0.0%
mv	21	943	1.0%	58.7%	0.6%	0.0%
rsh	22	934	1.0%	59.8%	0.3%	0.1%
clear	23	933	1.0%	60.8%	0.6%	0.0%
%	24	930	1.0%	61.8%	0.0%	0.0%
lpq	25	835	0.9%	62.7%	0.4%	0.0%
end	26	829	0.9%	63.6%	0.0%	0.0%
f	27	827	0.9%	64.6%	0.0%	0.0%
ruptime	28	824	0.9%	65.5%	0.5%	0.0%
msgs	29	784	0.9%	66.3%	0.3%	0.0%
dirs	30	774	0.9%	67.2%	0.2%	0.0%
ipq	31	761	0.8%	68.0%	0.0%	0.0%
set	32	696	0.8%	68.8%	0.5%	0.0%
make	33	627	0.7%	69.5%	0.2%	0.2%
rwho	34	606	0.7%	70.1%	0.4%	0.0%
pi	35	602	0.7%	70.8%	0.1%	0.1%
man	36	576	0.6%	71.4%	0.0%	0.0%
pushd	37	564	0.6%	72.1%	0.5%	0.0%
who	38	546	0.6%	72.7%	0.0%	0.0%
cp	39	540	0.6%	73.3%	0.2%	0.0%
from	40	517	0.6%	73.8%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (2)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
egrep	41	498	0.5%	74.4%	0.3%	0.0%
bg	42	497	0.5%	74.9%	0.0%	0.0%
source	43	492	0.5%	75.5%	0.4%	0.0%
rcp	44	475	0.5%	76.0%	0.1%	0.1%
chdir	45	473	0.5%	76.5%	0.5%	0.0%
ditroff	46	466	0.5%	77.0%	0.1%	0.4%
kill	47	459	0.5%	77.5%	0.2%	0.0%
df	48	453	0.5%	78.0%	0.0%	0.0%
Mail	49	439	0.5%	78.5%	0.3%	0.0%
lpr	50	409	0.5%	79.0%	0.1%	0.1%
while	51	402	0.4%	79.4%	0.0%	0.0%
date	52	368	0.4%	79.8%	0.0%	0.0%
history	53	344	0.4%	80.2%	0.3%	0.0%
talk	54	335	0.4%	80.6%	0.0%	0.0%
w	55	335	0.4%	80.9%	0.0%	0.0%
obj	56	332	0.4%	81.3%	0.0%	0.0%
rwhoson	57	331	0.4%	81.7%	0.1%	0.0%
sccs	58	323	0.4%	82.0%	0.0%	0.0%
0	59	316	0.3%	82.4%	0.0%	0.0%
uptime	60	288	0.3%	82.7%	0.1%	0.0%
cc	61	286	0.3%	83.0%	0.0%	0.0%
if	62	284	0.3%	83.3%	0.3%	0.0%
tail	63	259	0.3%	83.6%	0.1%	0.0%
ln	64	255	0.3%	83.9%	0.0%	0.0%
tar	65	224	0.2%	84.1%	0.0%	0.0%
alias	66	217	0.2%	84.4%	0.1%	0.0%
robots	67	215	0.2%	84.6%	0.0%	0.0%
fgrep	68	214	0.2%	84.8%	0.2%	0.0%
itroff	69	202	0.2%	85.0%	0.1%	0.2%
mc	70	202	0.2%	85.3%	0.0%	0.0%
page	71	201	0.2%	85.5%	0.0%	0.0%
chmod	72	200	0.2%	85.7%	0.0%	0.0%
last	73	200	0.2%	85.9%	0.0%	0.0%
mkdir	74	199	0.2%	86.2%	0.0%	0.0%
popd	75	198	0.2%	86.4%	0.2%	0.0%
sendmail	76	184	0.2%	86.6%	0.0%	0.0%
sleep	77	183	0.2%	86.8%	0.1%	0.1%
a.out	78	173	0.2%	87.0%	0.0%	0.0%
show	79	165	0.2%	87.1%	0.0%	0.0%
su	80	165	0.2%	87.3%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (3)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
q	81	164	0.2%	87.5%	0.0%	0.0%
du	82	157	0.2%	87.7%	0.0%	0.0%
idle	83	157	0.2%	87.9%	0.0%	0.0%
wc	84	150	0.2%	88.0%	0.1%	0.0%
head	85	149	0.2%	88.2%	0.1%	0.0%
nroff	86	146	0.2%	88.3%	0.0%	0.0%
n	87	144	0.2%	88.5%	0.0%	0.0%
dpq	88	143	0.2%	88.7%	0.0%	0.0%
telnet	89	143	0.2%	88.8%	0.0%	0.0%
csh	90	142	0.2%	89.0%	0.0%	0.0%
setenv	91	135	0.1%	89.1%	0.0%	0.0%
yank	92	132	0.1%	89.3%	0.0%	0.0%
@	93	127	0.1%	89.4%	0.0%	0.0%
luk	94	125	0.1%	89.5%	0.0%	0.0%
px	95	120	0.1%	89.7%	0.0%	0.0%
inc	96	119	0.1%	89.8%	0.0%	0.0%
rogue	97	118	0.1%	89.9%	0.0%	0.0%
l	98	117	0.1%	90.1%	0.0%	0.0%
next	99	113	0.1%	90.2%	0.0%	0.0%
suspend	100	112	0.1%	90.3%	0.1%	0.0%
tbl	101	109	0.1%	90.4%	0.0%	0.1%
go	102	101	0.1%	90.6%	0.0%	0.0%
ftp	103	100	0.1%	90.7%	0.0%	0.0%
whom	104	100	0.1%	90.8%	0.0%	0.0%
retrieve	105	96	0.1%	90.9%	0.1%	0.0%
stty	106	96	0.1%	91.0%	0.0%	0.0%
rmm	107	95	0.1%	91.1%	0.0%	0.0%
dl	108	90	0.1%	91.2%	0.1%	0.0%
unset	109	89	0.1%	91.3%	0.1%	0.0%
view	110	88	0.1%	91.4%	0.0%	0.0%
pdx	111	87	0.1%	91.5%	0.0%	0.0%
rmdir	112	85	0.1%	91.6%	0.0%	0.0%
diff	113	80	0.1%	91.7%	0.0%	0.0%
file	114	77	0.1%	91.7%	0.0%	0.0%
pic	115	77	0.1%	91.8%	0.0%	0.0%
whoami	116	77	0.1%	91.9%	0.0%	0.0%
scan	117	73	0.1%	92.0%	0.0%	0.0%
ucbvax	118	73	0.1%	92.1%	0.0%	0.0%
ping	119	70	0.1%	92.2%	0.0%	0.0%
write	120	69	0.1%	92.2%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (4)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
time	121	68	0.1%	92.3%	0.0%	0.0%
vpq	122	67	0.1%	92.4%	0.1%	0.0%
fing	123	61	0.1%	92.4%	0.0%	0.0%
pr	124	60	0.1%	92.5%	0.0%	0.0%
rrn	125	60	0.1%	92.6%	0.0%	0.0%
checknr	126	59	0.1%	92.6%	0.1%	0.0%
look	127	59	0.1%	92.7%	0.0%	0.0%
printenv	128	59	0.1%	92.8%	0.0%	0.0%
whereis	129	59	0.1%	92.8%	0.0%	0.0%
j	130	58	0.1%	92.9%	0.0%	0.0%
netstat	131	58	0.1%	93.0%	0.0%	0.0%
biff	132	57	0.1%	93.0%	0.0%	0.0%
rdist	133	55	0.1%	93.1%	0.0%	0.0%
rsl	134	55	0.1%	93.1%	0.0%	0.0%
test	135	55	0.1%	93.2%	0.0%	0.0%
r	136	54	0.1%	93.3%	0.0%	0.0%
client	137	53	0.1%	93.3%	0.0%	0.0%
Spell	138	52	0.1%	93.4%	0.0%	0.0%
radx	139	52	0.1%	93.4%	0.0%	0.0%
fortune	140	51	0.1%	93.5%	0.0%	0.0%
sort	141	51	0.1%	93.6%	0.0%	0.0%
tty	142	50	0.1%	93.6%	0.0%	0.0%
*	143	49	0.1%	93.7%	0.0%	0.0%
cas	144	49	0.1%	93.7%	0.0%	0.0%
deqn	145	49	0.1%	93.8%	0.0%	0.0%
find	146	49	0.1%	93.8%	0.0%	0.0%
nice	147	49	0.1%	93.9%	0.0%	0.0%
hm	148	48	0.1%	93.9%	0.0%	0.0%
i	149	48	0.1%	94.0%	0.0%	0.0%
NULL	150	47	0.1%	94.0%	0.0%	0.0%
eval	151	47	0.1%	94.1%	0.0%	0.0%
get	152	46	0.1%	94.1%	0.0%	0.0%
getNAME	153	45	0.0%	94.2%	0.0%	0.0%
on	154	44	0.0%	94.2%	0.0%	0.0%
soc	155	44	0.0%	94.3%	0.0%	0.0%
spell	156	41	0.0%	94.3%	0.0%	0.0%
2	157	40	0.0%	94.4%	0.0%	0.0%
lastcomm	158	40	0.0%	94.4%	0.0%	0.0%
t	159	40	0.0%	94.5%	0.0%	0.0%
at	160	38	0.0%	94.5%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal						(5)
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
mt	161	38	0.0%	94.5%	0.0%	0.0%
strings	162	38	0.0%	94.6%	0.0%	0.0%
dump	163	37	0.0%	94.6%	0.0%	0.0%
mlog	164	37	0.0%	94.7%	0.0%	0.0%
}	165	37	0.0%	94.7%	0.0%	0.0%
chfn	166	36	0.0%	94.7%	0.0%	0.0%
foreach	167	36	0.0%	94.8%	0.0%	0.0%
dbx	168	35	0.0%	94.8%	0.0%	0.0%
ia	169	35	0.0%	94.9%	0.0%	0.0%
lll-crg	170	35	0.0%	94.9%	0.0%	0.0%
ss	171	35	0.0%	94.9%	0.0%	0.0%
old_copy2.	172	34	0.0%	95.0%	0.0%	0.0%
whois	173	33	0.0%	95.0%	0.0%	0.0%
dd	174	31	0.0%	95.1%	0.0%	0.0%
repl	175	31	0.0%	95.1%	0.0%	0.0%
awk	176	30	0.0%	95.1%	0.0%	0.0%
print.xero	177	30	0.0%	95.2%	0.0%	0.0%
tee	178	30	0.0%	95.2%	0.0%	0.0%
'lpr'	179	29	0.0%	95.2%	0.0%	0.0%
chown	180	29	0.0%	95.2%	0.0%	0.0%
lprm	181	29	0.0%	95.3%	0.0%	0.0%
m	182	28	0.0%	95.3%	0.0%	0.0%
nm	183	28	0.0%	95.3%	0.0%	0.0%
pc	184	28	0.0%	95.4%	0.0%	0.0%
s	185	27	0.0%	95.4%	0.0%	0.0%
tset	186	27	0.0%	95.4%	0.0%	0.0%
OK	187	26	0.0%	95.5%	0.0%	0.0%
makewhatis	188	26	0.0%	95.5%	0.0%	0.0%
newesl	189	26	0.0%	95.5%	0.0%	0.0%
sh	190	26	0.0%	95.5%	0.0%	0.0%
yacc	191	26	0.0%	95.6%	0.0%	0.0%
eqn	192	25	0.0%	95.6%	0.0%	0.0%
initex	193	25	0.0%	95.6%	0.0%	0.0%
lpc	194	24	0.0%	95.7%	0.0%	0.0%
run	195	24	0.0%	95.7%	0.0%	0.0%
cmp	196	23	0.0%	95.7%	0.0%	0.0%
e	197	23	0.0%	95.7%	0.0%	0.0%
gprof	198	23	0.0%	95.8%	0.0%	0.0%
pw	199	23	0.0%	95.8%	0.0%	0.0%
rn	200	23	0.0%	95.8%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (6)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
apropos	201	22	0.0%	95.8%	0.0%	0.0%
y	202	22	0.0%	95.9%	0.0%	0.0%
"."	203	21	0.0%	95.9%	0.0%	0.0%
dc	204	21	0.0%	95.9%	0.0%	0.0%
eslrex	205	21	0.0%	95.9%	0.0%	0.0%
pwho	206	21	0.0%	96.0%	0.0%	0.0%
sed	207	21	0.0%	96.0%	0.0%	0.0%
unalias	208	21	0.0%	96.0%	0.0%	0.0%
{	209	21	0.0%	96.0%	0.0%	0.0%
fix_me	210	20	0.0%	96.0%	0.0%	0.0%
mine	211	20	0.0%	96.1%	0.0%	0.0%
prmail	212	20	0.0%	96.1%	0.0%	0.0%
script	213	20	0.0%	96.1%	0.0%	0.0%
systat	214	20	0.0%	96.1%	0.0%	0.0%
cal	215	19	0.0%	96.2%	0.0%	0.0%
dtbl	216	19	0.0%	96.2%	0.0%	0.0%
expn	217	19	0.0%	96.2%	0.0%	0.0%
grn	218	19	0.0%	96.2%	0.0%	0.0%
reset	219	19	0.0%	96.2%	0.0%	0.0%
restore	220	19	0.0%	96.3%	0.0%	0.0%
tn3270	221	19	0.0%	96.3%	0.0%	0.0%
touch	222	19	0.0%	96.3%	0.0%	0.0%
vial	223	19	0.0%	96.3%	0.0%	0.0%
`?	224	19	0.0%	96.3%	0.0%	0.0%
bye	225	18	0.0%	96.4%	0.0%	0.0%
iostat	226	18	0.0%	96.4%	0.0%	0.0%
mesg	227	18	0.0%	96.4%	0.0%	0.0%
pathalias	228	18	0.0%	96.4%	0.0%	0.0%
rup	229	18	0.0%	96.4%	0.0%	0.0%
!	230	17	0.0%	96.5%	0.0%	0.0%
ap.out	231	17	0.0%	96.5%	0.0%	0.0%
emacs	232	17	0.0%	96.5%	0.0%	0.0%
refer	233	17	0.0%	96.5%	0.0%	0.0%
b	234	16	0.0%	96.5%	0.0%	0.0%
d	235	16	0.0%	96.5%	0.0%	0.0%
modsite	236	16	0.0%	96.6%	0.0%	0.0%
mpx	237	16	0.0%	96.6%	0.0%	0.0%
mysh	238	16	0.0%	96.6%	0.0%	0.0%
rehash	239	16	0.0%	96.6%	0.0%	0.0%
ERROR	240	15	0.0%	96.6%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal						(7)
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
compress	241	15	0.0%	96.7%	0.0%	0.0%
i.out	242	15	0.0%	96.7%	0.0%	0.0%
initialize	243	15	0.0%	96.7%	0.0%	0.0%
mille	244	15	0.0%	96.7%	0.0%	0.0%
pd	245	15	0.0%	96.7%	0.0%	0.0%
rwhom	246	15	0.0%	96.7%	0.0%	0.0%
uu	247	15	0.0%	96.8%	0.0%	0.0%
vipw	248	15	0.0%	96.8%	0.0%	0.0%
ex	249	14	0.0%	96.8%	0.0%	0.0%
h	250	14	0.0%	96.8%	0.0%	0.0%
me	251	14	0.0%	96.8%	0.0%	0.0%
nc	252	14	0.0%	96.8%	0.0%	0.0%
p	253	14	0.0%	96.8%	0.0%	0.0%
rep	254	14	0.0%	96.9%	0.0%	0.0%
whoson	255	14	0.0%	96.9%	0.0%	0.0%
case	256	13	0.0%	96.9%	0.0%	0.0%
comp	257	13	0.0%	96.9%	0.0%	0.0%
vgrind	258	13	0.0%	96.9%	0.0%	0.0%
csrg	259	12	0.0%	96.9%	0.0%	0.0%
extern	260	12	0.0%	96.9%	0.0%	0.0%
nfontcheck	261	12	0.0%	97.0%	0.0%	0.0%
tex	262	12	0.0%	97.0%	0.0%	0.0%
tool	263	12	0.0%	97.0%	0.0%	0.0%
undump	264	12	0.0%	97.0%	0.0%	0.0%
window	265	12	0.0%	97.0%	0.0%	0.0%
x	266	12	0.0%	97.0%	0.0%	0.0%
LS	267	11	0.0%	97.0%	0.0%	0.0%
c	268	11	0.0%	97.0%	0.0%	0.0%
ccom	269	11	0.0%	97.1%	0.0%	0.0%
findit	270	11	0.0%	97.1%	0.0%	0.0%
groups	271	11	0.0%	97.1%	0.0%	0.0%
owd	272	11	0.0%	97.1%	0.0%	0.0%
pix	273	11	0.0%	97.1%	0.0%	0.0%
split	274	11	0.0%	97.1%	0.0%	0.0%
timedc	275	11	0.0%	97.1%	0.0%	0.0%
uncompress	276	11	0.0%	97.1%	0.0%	0.0%
unsetenv	277	11	0.0%	97.2%	0.0%	0.0%
whatis	278	11	0.0%	97.2%	0.0%	0.0%
]	279	10	0.0%	97.2%	0.0%	0.0%
break	280	10	0.0%	97.2%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal						(8)
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
dpcheck	281	10	0.0%	97.2%	0.0%	0.0%
ll	282	10	0.0%	97.2%	0.0%	0.0%
palias	283	10	0.0%	97.2%	0.0%	0.0%
passwd	284	10	0.0%	97.2%	0.0%	0.0%
prev	285	10	0.0%	97.2%	0.0%	0.0%
renice	286	10	0.0%	97.3%	0.0%	0.0%
rslcreate	287	10	0.0%	97.3%	0.0%	0.0%
which	288	10	0.0%	97.3%	0.0%	0.0%
`]	289	9	0.0%	97.3%	0.0%	0.0%
'lpq'	290	9	0.0%	97.3%	0.0%	0.0%
a	291	9	0.0%	97.3%	0.0%	0.0%
adb	292	9	0.0%	97.3%	0.0%	0.0%
bib	293	9	0.0%	97.3%	0.0%	0.0%
dpstart	294	9	0.0%	97.3%	0.0%	0.0%
equel	295	9	0.0%	97.3%	0.0%	0.0%
hello	296	9	0.0%	97.4%	0.0%	0.0%
mo	297	9	0.0%	97.4%	0.0%	0.0%
quiz	298	9	0.0%	97.4%	0.0%	0.0%
rdump	299	9	0.0%	97.4%	0.0%	0.0%
scalepx	300	9	0.0%	97.4%	0.0%	0.0%
sp-	301	9	0.0%	97.4%	0.0%	0.0%
LABEL	302	8	0.0%	97.4%	0.0%	0.0%
Z	303	8	0.0%	97.4%	0.0%	0.0%
ar	304	8	0.0%	97.4%	0.0%	0.0%
atq	305	8	0.0%	97.4%	0.0%	0.0%
co	306	8	0.0%	97.5%	0.0%	0.0%
didrs	307	8	0.0%	97.5%	0.0%	0.0%
fontcheck	308	8	0.0%	97.5%	0.0%	0.0%
gnumacs	309	8	0.0%	97.5%	0.0%	0.0%
igrind	310	8	0.0%	97.5%	0.0%	0.0%
sl	311	8	0.0%	97.5%	0.0%	0.0%
ul	312	8	0.0%	97.5%	0.0%	0.0%
uucp	313	8	0.0%	97.5%	0.0%	0.0%
&	314	7	0.0%	97.5%	0.0%	0.0%
".PP"	315	7	0.0%	97.5%	0.0%	0.0%
".sp"	316	7	0.0%	97.5%	0.0%	0.0%
atrm	317	7	0.0%	97.5%	0.0%	0.0%
dterm	318	7	0.0%	97.6%	0.0%	0.0%
f77	319	7	0.0%	97.6%	0.0%	0.0%
gn	320	7	0.0%	97.6%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (9)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
gomoku	321	7	0.0%	97.6%	0.0%	0.0%
inp.o	322	7	0.0%	97.6%	0.0%	0.0%
ireqn	323	7	0.0%	97.6%	0.0%	0.0%
job	324	7	0.0%	97.6%	0.0%	0.0%
jos	325	7	0.0%	97.6%	0.0%	0.0%
k	326	7	0.0%	97.6%	0.0%	0.0%
leave	327	7	0.0%	97.6%	0.0%	0.0%
lks	328	7	0.0%	97.6%	0.0%	0.0%
mai	329	7	0.0%	97.6%	0.0%	0.0%
mroe	330	7	0.0%	97.6%	0.0%	0.0%
named	331	7	0.0%	97.7%	0.0%	0.0%
opq	332	7	0.0%	97.7%	0.0%	0.0%
rstat	333	7	0.0%	97.7%	0.0%	0.0%
ru	334	7	0.0%	97.7%	0.0%	0.0%
sentinfo	335	7	0.0%	97.7%	0.0%	0.0%
sr	336	7	0.0%	97.7%	0.0%	0.0%
tip	337	7	0.0%	97.7%	0.0%	0.0%
uuencode	338	7	0.0%	97.7%	0.0%	0.0%
vmstat	339	7	0.0%	97.7%	0.0%	0.0%
^X	340	6	0.0%	97.7%	0.0%	0.0%
^]z	341	6	0.0%	97.7%	0.0%	0.0%
,	342	6	0.0%	97.7%	0.0%	0.0%
1	343	6	0.0%	97.7%	0.0%	0.0%
chgrp	344	6	0.0%	97.7%	0.0%	0.0%
ci	345	6	0.0%	97.8%	0.0%	0.0%
digrind	346	6	0.0%	97.8%	0.0%	0.0%
fish	347	6	0.0%	97.8%	0.0%	0.0%
jbos	348	6	0.0%	97.8%	0.0%	0.0%
lock	349	6	0.0%	97.8%	0.0%	0.0%
login	350	6	0.0%	97.8%	0.0%	0.0%
logut	351	6	0.0%	97.8%	0.0%	0.0%
lsf	352	6	0.0%	97.8%	0.0%	0.0%
parentg	353	6	0.0%	97.8%	0.0%	0.0%
sync	354	6	0.0%	97.8%	0.0%	0.0%
troff	355	6	0.0%	97.8%	0.0%	0.0%
umask	356	6	0.0%	97.8%	0.0%	0.0%
watchhost	357	6	0.0%	97.8%	0.0%	0.0%
xp	358	6	0.0%	97.8%	0.0%	0.0%
z	359	6	0.0%	97.8%	0.0%	0.0%
^H	360	5	0.0%	97.9%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (10						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
addcal	361	5	0.0%	97.9%	0.0%	0.0%
adventure	362	5	0.0%	97.9%	0.0%	0.0%
call	363	5	0.0%	97.9%	0.0%	0.0%
ccd	364	5	0.0%	97.9%	0.0%	0.0%
chsh	365	5	0.0%	97.9%	0.0%	0.0%
cle	366	5	0.0%	97.9%	0.0%	0.0%
comm	367	5	0.0%	97.9%	0.0%	0.0%
corbomite	368	5	0.0%	97.9%	0.0%	0.0%
delayben.c	369	5	0.0%	97.9%	0.0%	0.0%
folder	370	5	0.0%	97.9%	0.0%	0.0%
getpaper	371	5	0.0%	97.9%	0.0%	0.0%
hashstat	372	5	0.0%	97.9%	0.0%	0.0%
help	373	5	0.0%	97.9%	0.0%	0.0%
idi	374	5	0.0%	97.9%	0.0%	0.0%
iq	375	5	0.0%	97.9%	0.0%	0.0%
jj	376	5	0.0%	97.9%	0.0%	0.0%
ld	377	5	0.0%	97.9%	0.0%	0.0%
lo	378	5	0.0%	98.0%	0.0%	0.0%
lss	379	5	0.0%	98.0%	0.0%	0.0%
m.out	380	5	0.0%	98.0%	0.0%	0.0%
notify	381	5	0.0%	98.0%	0.0%	0.0%
oparentg	382	5	0.0%	98.0%	0.0%	0.0%
postnews	383	5	0.0%	98.0%	0.0%	0.0%
printl	384	5	0.0%	98.0%	0.0%	0.0%
probabilit	385	5	0.0%	98.0%	0.0%	0.0%
pwdd	386	5	0.0%	98.0%	0.0%	0.0%
runss	387	5	0.0%	98.0%	0.0%	0.0%
smtp	388	5	0.0%	98.0%	0.0%	0.0%
sortm	389	5	0.0%	98.0%	0.0%	0.0%
stat	390	5	0.0%	98.0%	0.0%	0.0%
strip	391	5	0.0%	98.0%	0.0%	0.0%
sysline	392	5	0.0%	98.0%	0.0%	0.0%
tree	393	5	0.0%	98.0%	0.0%	0.0%
v	394	5	0.0%	98.0%	0.0%	0.0%
vcp	395	5	0.0%	98.0%	0.0%	0.0%
virtex	396	5	0.0%	98.1%	0.0%	0.0%
webster	397	5	0.0%	98.1%	0.0%	0.0%
wpd	398	5	0.0%	98.1%	0.0%	0.0%
zmore	399	5	0.0%	98.1%	0.0%	0.0%
`L	400	4	0.0%	98.1%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal						(11)
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
#include	401	4	0.0%	98.1%	0.0%	0.0%
".ds"	402	4	0.0%	98.1%	0.0%	0.0%
10	403	4	0.0%	98.1%	0.0%	0.0%
V.2.2	404	4	0.0%	98.1%	0.0%	0.0%
[	405	4	0.0%	98.1%	0.0%	0.0%
addbib	406	4	0.0%	98.1%	0.0%	0.0%
al	407	4	0.0%	98.1%	0.0%	0.0%
ca	408	4	0.0%	98.1%	0.0%	0.0%
continue	409	4	0.0%	98.1%	0.0%	0.0%
ctags	410	4	0.0%	98.1%	0.0%	0.0%
daemon	411	4	0.0%	98.1%	0.0%	0.0%
ddirs	412	4	0.0%	98.1%	0.0%	0.0%
dgr	413	4	0.0%	98.1%	0.0%	0.0%
dgs	414	4	0.0%	98.1%	0.0%	0.0%
fetch	415	4	0.0%	98.1%	0.0%	0.0%
figner	416	4	0.0%	98.1%	0.0%	0.0%
fn	417	4	0.0%	98.1%	0.0%	0.0%
forw	418	4	0.0%	98.2%	0.0%	0.0%
glob	419	4	0.0%	98.2%	0.0%	0.0%
gre	420	4	0.0%	98.2%	0.0%	0.0%
hostname	421	4	0.0%	98.2%	0.0%	0.0%
int	422	4	0.0%	98.2%	0.0%	0.0%
ip	423	4	0.0%	98.2%	0.0%	0.0%
kj	424	4	0.0%	98.2%	0.0%	0.0%
lbl-csam	425	4	0.0%	98.2%	0.0%	0.0%
ll.bib	426	4	0.0%	98.2%	0.0%	0.0%
lpiq	427	4	0.0%	98.2%	0.0%	0.0%
mal	428	4	0.0%	98.2%	0.0%	0.0%
mbox	429	4	0.0%	98.2%	0.0%	0.0%
mount	430	4	0.0%	98.2%	0.0%	0.0%
nray	431	4	0.0%	98.2%	0.0%	0.0%
nta-vax	432	4	0.0%	98.2%	0.0%	0.0%
out	433	4	0.0%	98.2%	0.0%	0.0%
pargv	434	4	0.0%	98.2%	0.0%	0.0%
printi	435	4	0.0%	98.2%	0.0%	0.0%
prompt.wul	436	4	0.0%	98.2%	0.0%	0.0%
pwdu	437	4	0.0%	98.2%	0.0%	0.0%
query	438	4	0.0%	98.2%	0.0%	0.0%
rmf	439	4	0.0%	98.2%	0.0%	0.0%
robtest	440	4	0.0%	98.2%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (12						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
rsl2	441	4	0.0%	98.3%	0.0%	0.0%
ruptie	442	4	0.0%	98.3%	0.0%	0.0%
rwhoso	443	4	0.0%	98.3%	0.0%	0.0%
smgs	444	4	0.0%	98.3%	0.0%	0.0%
stop	445	4	0.0%	98.3%	0.0%	0.0%
switch	446	4	0.0%	98.3%	0.0%	0.0%
uncompact	447	4	0.0%	98.3%	0.0%	0.0%
wait	448	4	0.0%	98.3%	0.0%	0.0%
wd	449	4	0.0%	98.3%	0.0%	0.0%
wump	450	4	0.0%	98.3%	0.0%	0.0%
zork	451	4	0.0%	98.3%	0.0%	0.0%
!	452	3	0.0%	98.3%	0.0%	0.0%
"	453	3	0.0%	98.3%	0.0%	0.0%
".nr"	454	3	0.0%	98.3%	0.0%	0.0%
411	455	3	0.0%	98.3%	0.0%	0.0%
6	456	3	0.0%	98.3%	0.0%	0.0%
>	457	3	0.0%	98.3%	0.0%	0.0%
Appendix	458	3	0.0%	98.3%	0.0%	0.0%
CD	459	3	0.0%	98.3%	0.0%	0.0%
FG	460	3	0.0%	98.3%	0.0%	0.0%
LA	461	3	0.0%	98.3%	0.0%	0.0%
TERM	462	3	0.0%	98.3%	0.0%	0.0%
ZZ	463	3	0.0%	98.3%	0.0%	0.0%
ZZZ	464	3	0.0%	98.3%	0.0%	0.0%
accessdb	465	3	0.0%	98.3%	0.0%	0.0%
arp	466	3	0.0%	98.3%	0.0%	0.0%
cd..	467	3	0.0%	98.3%	0.0%	0.0%
cl	468	3	0.0%	98.4%	0.0%	0.0%
comm.file	469	3	0.0%	98.4%	0.0%	0.0%
cory-ec	470	3	0.0%	98.4%	0.0%	0.0%
create	471	3	0.0%	98.4%	0.0%	0.0%
dal	472	3	0.0%	98.4%	0.0%	0.0%
delta	473	3	0.0%	98.4%	0.0%	0.0%
dfl	474	3	0.0%	98.4%	0.0%	0.0%
dirstats	475	3	0.0%	98.4%	0.0%	0.0%
dist_to_lo	476	3	0.0%	98.4%	0.0%	0.0%
emp	477	3	0.0%	98.4%	0.0%	0.0%
eslcreate	478	3	0.0%	98.4%	0.0%	0.0%
fc	479	3	0.0%	98.4%	0.0%	0.0%
fmt	480	3	0.0%	98.4%	0.0%	0.0%

Table 26 - Five Hundred Command Totals from Terminal (13)						
Command	Rank	Overall Frequency	Per Cent of All Commands	Cumulative Per Cent	Alias Per Cent	Background Per Cent
g	481	3	0.0%	98.4%	0.0%	0.0%
googoo	482	3	0.0%	98.4%	0.0%	0.0%
hh	483	3	0.0%	98.4%	0.0%	0.0%
htable	484	3	0.0%	98.4%	0.0%	0.0%
hunt	485	3	0.0%	98.4%	0.0%	0.0%
ippq	486	3	0.0%	98.4%	0.0%	0.0%
jbs	487	3	0.0%	98.4%	0.0%	0.0%
jk	488	3	0.0%	98.4%	0.0%	0.0%
josb	489	3	0.0%	98.4%	0.0%	0.0%
kil	490	3	0.0%	98.4%	0.0%	0.0%
kl	491	3	0.0%	98.4%	0.0%	0.0%
lcd	492	3	0.0%	98.4%	0.0%	0.0%
lds	493	3	0.0%	98.4%	0.0%	0.0%
learn	494	3	0.0%	98.4%	0.0%	0.0%
letters	495	3	0.0%	98.4%	0.0%	0.0%
lex	496	3	0.0%	98.4%	0.0%	0.0%
lf	497	3	0.0%	98.4%	0.0%	0.0%
lisp	498	3	0.0%	98.5%	0.0%	0.0%
lk	499	3	0.0%	98.5%	0.0%	0.0%
lookpsych	500	3	0.0%	98.5%	0.0%	0.0%
Total	—	90774	100.0%	100.0%	31.4%	2.1%

4.2.6. Detailed Accounting Statistics

Accounting statistics for the one hundred commands that appear most frequently in the accounting file are given in Table 27, "Accounting Statistics for 100 Commands." Note that the order of the commands in this table bears little resemblance to the order of the commands in Table 26, which ranks commands by their usage at the terminal. Note also that the most frequently used commands use relatively little CPU time.

Table 27 - Accounting Statistics for 100 Commands (1)

Command	Rank	Accounting Frequency	Average System Time	Average User Time	Average Elapsed Time	Average Memory Usage	Average I/O Operations
sendmail	1	22045	0:00.4	0:00.0	0:15.3	155	22
lpd	2	16222	0:00.2	0:00.0	0:23.9	46	14
csh	3	15181	0:02.2	0:00.5	39:08.2	133	82
ls	4	10993	0:00.2	0:00.0	0:01.3	23	6
more	5	7697	0:00.2	0:00.0	0:37.8	24	4
sh	6	7406	0:00.1	0:00.0	1:06.8	20	4
echo	7	7091	0:00.0	0:00.0	0:05.7	14	3
mail	8	6449	0:00.6	0:00.3	1:53.7	54	34
rm	9	5812	0:00.0	0:00.0	0:00.8	7	8
vi	10	5794	0:02.3	0:02.5	7:31.9	125	35
cat	11	5596	0:00.1	0:00.0	0:35.7	38	5
date	12	5487	0:00.1	0:00.0	0:03.2	30	3
test	13	3453	0:00.0	0:00.0	0:00.2	9	3
rlogin	14	3444	0:04.3	0:00.0	41:50.9	22	5
stty	15	3144	0:00.0	0:00.0	0:00.2	7	0
grep	16	2974	0:00.2	0:00.0	0:02.3	51	8
lpq	17	2811	0:00.2	0:00.0	0:02.7	25	11
tset	18	2612	0:00.3	0:00.0	0:27.7	34	10
ips	19	2465	0:02.6	0:02.6	1:38.8	31	13
who	20	2421	0:00.0	0:00.0	0:28.0	14	2
msgs	21	2263	0:00.5	0:00.0	0:59.2	25	19
sed	22	2249	0:00.1	0:00.0	0:05.1	59	5
hostname	23	2230	0:00.0	0:00.0	0:00.3	13	1
clear	24	2228	0:00.0	0:00.0	0:00.4	7	2
comsat	25	1936	0:00.1	0:00.0	2:34.1	24	5
sleep	26	1931	0:00.0	0:00.0	1:24.9	2	0
ps	27	1796	0:00.6	0:00.0	0:06.8	61	7
finger	28	1795	0:00.3	0:00.0	0:05.4	30	12
mv	29	1752	0:00.0	0:00.0	0:00.5	6	7
cpp	30	1610	0:00.2	0:00.0	0:02.5	38	18
cc	31	1593	0:00.1	0:00.0	0:37.4	12	10
rlogind	32	1454	0:14.7	0:00.3	53:50.4	19	16
la	33	1387	0:00.1	0:00.0	0:00.6	13	9
pwd	34	1366	0:00.1	0:00.0	0:00.5	9	8
lpr	35	1289	0:00.6	0:00.0	2:47.0	30	46
ruptime	36	1243	0:00.2	0:00.0	0:03.2	14	32
dit-imagen	37	1238	0:01.5	0:04.0	0:27.2	113	88
u	38	1173	0:00.0	0:00.0	0:00.4	11	2
ccom	39	1129	0:01.2	0:05.2	0:35.2	197	6
atrun	40	1082	0:00.1	0:00.0	0:12.5	10	14

Table 27 - Accounting Statistics for 100 Commands (2)							
Command	Rank	Accounting Frequency	Average System Time	Average User Time	Average Elapsed Time	Average Memory Usage	Average I/O Operations
rsh	41	1063	0:00.2	0:00.0	0:43.6	21	4
fortune	42	1018	0:00.1	0:00.0	0:08.7	9	9
as	43	984	0:01.7	0:05.9	0:26.0	92	33
egrep	44	929	0:00.2	0:00.9	0:06.4	28	8
getty	45	912	0:00.1	0:00.0	7:19.5	13	3
rcp	46	902	0:01.2	0:00.0	0:16.7	26	38
catdvi	47	870	0:01.1	0:04.0	0:27.5	42	35
dviimp	48	855	0:01.8	0:07.1	0:44.4	79	58
f	49	813	0:00.2	0:00.0	0:11.8	26	7
px	50	741	0:00.8	0:02.1	1:46.4	45	9
[	51	720	0:00.0	0:00.0	0:00.1	18	1
fingerd	52	697	0:00.1	0:00.0	0:11.2	24	6
cp	53	688	0:00.2	0:00.0	0:01.3	9	28
tty	54	676	0:00.0	0:00.0	0:05.3	31	4
sysline	55	670	0:07.8	0:01.7	31:59.2	48	191
login	56	649	0:00.7	0:00.0	17:35.2	19	12
rwho	57	648	0:00.4	0:00.0	0:08.8	22	29
pi	58	634	0:00.9	0:00.5	0:15.2	99	43
uptime	59	631	0:00.2	0:00.0	0:01.3	28	10
from	60	630	0:00.1	0:00.0	0:00.8	14	6
make	61	600	0:00.7	0:00.0	4:54.7	43	28
troff_p	62	546	0:06.2	0:36.2	4:17.7	130	99
man	63	538	0:00.2	0:00.0	0:51.1	12	14
name	64	530	0:00.0	0:00.0	0:00.3	51	4
biff	65	506	0:00.0	0:00.0	0:00.2	5	1
rwhoson	66	494	0:00.5	0:00.0	0:04.4	25	34
pdx	67	439	0:01.2	0:00.2	2:44.0	49	16
sort	68	436	0:00.2	0:00.2	0:19.4	20	11
w	69	435	0:01.9	0:00.0	1:33.6	60	13
ld	70	435	0:01.4	0:00.5	0:11.2	101	118
talkd	71	384	0:00.1	0:00.0	1:35.3	25	5
df	72	371	0:00.3	0:00.0	0:02.0	17	54
Mail	73	353	0:01.2	0:00.8	2:27.9	84	67
file	74	345	0:00.0	0:00.0	0:00.3	7	2
rshd	75	327	0:00.3	0:00.0	0:33.9	27	14
mc	76	325	0:00.0	0:00.0	0:00.6	7	3
tail	77	312	0:00.1	0:00.0	0:07.7	12	5
robots	78	307	0:03.0	0:07.0	2:42.6	56	10
ln	79	302	0:00.1	0:00.0	0:00.4	12	9
talk	80	299	0:06.9	0:01.5	4:48.7	51	8

Table 27 - Accounting Statistics for 100 Commands (3)							
Command	Rank	Accounting Frequency	Average System Time	Average User Time	Average Elapsed Time	Average Memory Usage	Average I/O Operations
syslogd	81	283	0:01.1	0:00.1	0:00.0	25	64
mesg	82	279	0:00.0	0:00.0	0:00.3	4	1
inc	83	279	0:00.5	0:00.1	0:04.1	38	39
c2	84	279	0:00.5	0:01.5	0:10.3	41	8
troff	85	276	0:04.7	0:26.7	3:27.7	100	77
xstr	86	270	0:00.4	0:00.0	0:07.7	37	13
chmod	87	267	0:00.0	0:00.0	0:00.3	5	3
diff	88	257	0:02.3	0:12.3	2:13.1	39	18
mkdir	89	253	0:00.0	0:00.0	0:00.5	3	10
fgrep	90	232	0:00.1	0:00.0	0:30.0	8	7
dd	91	230	0:12.8	0:38.3	5:28.2	15	4
a.out	92	225	0:02.1	0:10.7	4:31.6	18	4
tar	93	219	0:04.0	0:02.7	0:49.3	20	106
page	94	218	0:00.3	0:00.0	1:21.8	25	3
du	95	216	0:01.3	0:00.0	0:07.7	14	85
rmt	96	205	0:37.5	0:00.6	16:13.9	17	2
last	97	198	0:00.5	0:01.0	0:15.8	21	21
idle	98	196	0:00.2	0:00.0	0:02.8	14	2
find	99	193	0:25.7	0:06.4	1:45.1	15	1805
spell	100	192	0:00.4	0:00.2	0:31.5	55	20

4.2.7. Comparisons to Previous Work

[Rosson84] found that programmers were more likely than secretaries to use complex features. My results in Table 7 showing how often alias substitutions are used support this conclusion. [Penniman84] shows an average of 53.7 transactions per session, while Table 15 shows an average of 21 commands per login session. Note, however, that my data is biased by "misc-non-humans," who never type commands at their terminals. Recall, also, that commands in *.login* files, *.cshrc* files, and *.logout* files are executed each time a user logs in, but are not included in the average number of commands per login session. If one totals the average number of commands in a *.login* file (12), a *.cshrc* file (21), a *.logout* file (2), and a login session (21), as shown in Table 14, the result (56) is quite close to Penniman's result.

[Hanson84] shows the twenty most frequently used Unix commands, using command-line data. They include pipes, sequential execution, input/output redirection, and background symbols in their commands. Since I do not include these symbols, I will compare the other sixteen commands in their data to the top sixteen commands in Table 26. My comparison is shown in Table 28, "Comparison of Command Ranks." A null entry in the third column of the following table indicates that this command was not one of the 500 commands listed in Table 26.

Table 28 Comparison of Command Ranks		
Command	Rank in [Hanson84]	Rank in Table 26
cd	1	3
ls	2	1
cat	3	14
vi	4	2
ed	5	—
rm	6	10
Mail	7	49
nroff	8	86
mail	9	8
mv	10	21
grep	11	19
col	12	—
echo	13	7
tail	14	63
pwd	15	20
awk	16	176

[Hanson84] stated that the majority of their users were engaged in document preparation, which explains most of the differences in our results. The *nroff* program is a text formater. *col* program filters control characters from text, and the *awk* program is a pattern scanning and processing language. All three of these programs are more likely to be used in document preparation than in programming. The *Mail* program is just another version of the *mail* program.

5. Performance Evaluation

While tracing, the C Shell uses additional CPU time, additional input/output time, and additional disk space, the actual impact on the system was minimal. Only a few lines of code were added to the C Shell, and, as explained in Section 3.1.1, system calls were kept to a minimum, so the CPU overhead was small. In most cases, the only system calls were the write system calls, which were only executed after at least 1024 bytes of data had accumulated. Minimizing the number of write system calls also minimized the input/output overhead, as a few large writes are more efficient than many small writes. The impact of the additional disk usage was kept to a minimum by frequently moving the trace file to tape.

6. Lessons Learned

Designing and implementing a program becomes much easier after the program has been designed and implemented. Several mistakes were made during the design and analysis of my programs.

6.1. Design Mistakes

Designing and implementing the program that combined the C Shell data with the accounting data took well over half of the total design and implementation time, as a result of many problems. One problem was that the accounting file includes commands from the users "root" and "daemon," which are often executed at the terminal and do not correspond to any trace data. Another problem was that accounting is "turned off" when the file

system that contains the accounting file is getting full. The worst problems, however, were associated with trying to process commands, such as *make* and *sccs*, that generate multiple accounting records. The algorithm to match these commands to accounting file commands became quite complex. In addition, the resulting records became quite large and difficult to manage. Some records were several thousand bytes long. Perhaps the time spent trying to combine the C Shell data with the accounting data would have been better spent tracing the Bourne *(sh)* shell, so that the data for scripts would have been complete. The accounting data could then be analyzed independently, without combining it with the C Shell data.

6.2. Implementation Lessons

Processing large quantities of data caused several problems. There was not enough disk space to hold all the data being processed, so the data had to be put on tape and copied back onto disk in sections for processing. These problems were simplified by breaking each file into smaller pieces, with one file for each day of data. Another advantage of breaking the data into smaller pieces was that a programming bug could be isolated to the particular day of data causing the bug. Then the program could be debugged using just that day of data.

Two problems resulted from the data formats. One problem resulted from storing the data in binary instead of ASCII, and the other resulted from using one or two bytes to store integers instead of using four bytes. Storing the data in binary instead of ASCII form was, by far, my worst mistake. All data files, except the initial file of C Shell trace data were stored in binary, to conserve space. Integers stored in binary can be stored in one, two, or four byte fields, while integers stored in ASCII will require one byte for each digit. So, I decided to store the integers in binary form. The result was that I could not edit the data files, or print their contents, making debugging extremely difficult. The second problem associated with the data formats was again a result of trying to conserve space. Integers that were expected to be quite small were stored in one or two byte fields. Inevitably, the unexpected would happen, and an overflow would occur, creating strange results. These problems were also difficult to debug, as they were caused by only a few records, making it difficult to track down the problems.

Another minor design error was my method for the creation of login session identifiers. As explained in Section 3.4.3, my login session identifiers were created during post-processing from *csh* accounting records. A better, more accurate method would have been to create the login session identifiers directly within the C Shell tracer, while the data was being recorded.

7. Conclusion

Most results were not surprising. Usage patterns are usually simple. A few commands account for the vast majority of usage, as found in earlier studies. Most command lines contain only one command. Users often use aliases and history substitutions to decrease the number of keys they must type. CPU-intensive commands are often executed in the background. Complex features are rarely used.

One result was very surprising, however. Almost 30 per cent of the data was generated by the system using the C Shell to execute command files.

8. Acknowledgements

I would like to thank my research advisor, Alan Jay Smith, for his support during this project. I would also like to thank Larry Rowe for being my second reader. Thanks are also due to Jim Bloom, a member of the Computer Systems Research Group, for providing answers to my many questions. In addition, I would like to give special thanks to Mike

Karels, another member of the Computer Systems Research Group, who allowed me to collect the necessary data and provided answers to what must have seemed an endless stream of questions.

References

[Boies74]
Boies, S.J., "User Behavior on an Interactive Computer System," *IBM Systems Journal*, 1974, Vol. 13, No. 1, pp. 2-17.

[Bourne78]
Bourne, S.R., "The Unix Time-Sharing System: The Unix Shell," *The Bell System Technical Journal*, July-August, 1978, Vol. 57, No. 6, pp. 1971-1990.

[Computer84a]
Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, *UNIX User's Manual: Reference Guide*, 2 vols. Berkeley, California, March 1984.

[Computer84b]
Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, *UNIX User's Manual: Supplementary Documents*, 2 vols. Berkeley, California, March 1984.

[Hanson84]
Hanson, Stephen Jose, Kraut, Robert E., and Farber, James M., "Interface Design and Multivariate Analysis of UNIX Command Use," *ACM Transactions on Office Information Systems*, Vol. 2, No. 1, March 1984, pp. 42-57.

[Kernighan78]
Kernighan, Brian W. and Ritchie, Dennis M., *The C Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978.

[Kraut83]
Kraut, Robert E., Hanson, Stephen J., and Farber, James M., "Command Use and Interface Design" *Proceedings CHI'83 Human Factors in Computing Systems*, (Boston, Massachusetts, December 13-15), 1983, ACM, New York.

[Kurihara84]
Kurihara, Motoshi, Kikuchi, Kiyoshi, Iwai, Isamu, Saitou, Mitsuo, and Doi, Miwako, "An Office Workstation with a More Friendly User Interface for Document Editing," *Proceedings Compcon Fall '84*, (Arlington, Virginia, September 16-20), 1984, IEEE Computer Society, Computer Society Press.

[Lambert84]
Lambert, G.N. "A Comparative Study of System Response Time on Program Developer Productivity," *IBM Systems Journal*, 1984, Vol. 23, No. 1, pp. 36-43.

[Maeda84]
Maeda, K., Miyake, Y., Nievergelt, J., and Saito, Y., "A Comparative Study of Man-Machine Interfaced in Interactive Systems," *SIGCHI Bulletin*, October 1984, ACM, New York.

[McKusick84]
McKusick, Marshall, Joy, William, Leffler, Samuel, and Fabry, Robert, "A Fast File System for UNIX," *ACM Transactions on Computer Systems*, August 1984, Vol. 2, No. 3, pp. 181-197.

[Penniman84]
Penniman, W. David, "A Methodology for Evaluating Interactive System Usage," *SIGCHI Bulletin*, April, 1984, ACM, New York.

[Ritchie74]
Ritchie, Dennis M. and Thompson, Ken, "The UNIX Time-Sharing System," *Communications of the ACM*, July, 1974, Vol. 17, No. 7, pp. 365-375.

[Ritchie78]
Ritchie, D.M., "UNIX Time-Sharing System: A Retrospective," *The Bell System Technical Journal*, July-August 1978, Vol. 57, No. 6, pp. 1947-1969.

[Rosson84]
Rosson, Mary Beth, "Effects of Experience on Learning, Using, and Evaluating a Text Editor," *Human Factors*, Vol. 26, No. 4, 1984, pp. 463-475.

[Thadhani81]
Thadhani, A.J., "Interactive User Productivity," *IBM Systems Journal*, Vol. 20, No. 4, 1984, pp. 407-423.

[Thadhani84]
Thadhani, A.J., "Factors Affecting Programmer Productivity During Application Development," *IBM Systems Journal*, Vol. 23, No. 1, 1984, pp. 19-35.

Appendix A

C Shell Features

The features of the C Shell that are analyzed in this paper are briefly discussed below. The intent is to give the reader a brief understanding of what is being analyzed. The intent is NOT to provide a tutorial or a thorough understanding of the features discussed. If the reader desires a more thorough understanding, consult the *csh* manual entry in [Computer84a] or "An Introduction to the C Shell" in [Computer84b].

Built-in Commands: Certain commands are *built-in* to the C Shell. These commands are executed directly by the C Shell without calling another program. These commands are usually simple commands, such as, *alias* (explained below), *cd* (change directory), or *logout* (log off), Since these commands are executed directly by the C Shell, they do not have accounting records in the accounting file. Any CPU or input/output time used to execute these commands will be included in the CPU time or input/output time for the accounting record for the C Shell *(csh)*.

Job Control - Background/Foreground - %: The Unix C Shell has *job control*. This allows a user to run programs in the *background*, so that the user may run another program, such as the editor, in the *foreground*. Typing a "&" at the end of any command line will cause that command line to be executed in the background. Users may, at any time, stop a job running in the foreground by typing <ctrl> Z. The user may then re-start the job in the background or the foreground. The job can be re-started in the foreground by typing *fg*, and it can be re-started in the background by typing *bg*. The "%" character can also be used to bring a job running in the background into the foreground.

Alias Substitutions: Users may define *aliases*. Aliases are usually used when the user wishes to substitute a short character string for a longer character string. For example, *more* is a program that prints files on the user's terminal screen. Often, users will *alias* m to more. So, instead of typing *more filename*, the user may type *m filename*. The user defines this alias by typing *alias m more* or putting this command line in his *.cshrc* file or his *.login* file. Aliases are often much more complex than this example. Users commonly declare many aliases in their *.cshrc* file or their *.login* file. (*.cshrc* and *.login* files are explained below.)

History Substitutions: History substitutions allow a user to repeat part or all of a previous command line. For instance, if a user types "!!," the previous command line will be executed again. If a user types "!c," the last command line that began with the character c will be executed again. More complex substitutions allow a user to modify a previous command line. before executing it again.

.login, .cshrc, .logout, and Other Command Files: Command files contain command lines. A user may type *source filename*, and the command lines in *filename* will be executed. The *.login* file, *.cshrc* file, and *.logout* file are special command files. The command lines in the *.login* and *.cshrc* files are executed every time the user logs in. The commands in the *.cshrc* file are executed under other circumstances, as well. not relevant to this discussion. The command lines in the *.logout* file are executed every time the user logs out.

Scripts: Scripts are similar to command files. If a user simply types *filename*, the command lines in *filename* will be executed.

Syntax Errors: A *syntax error* occurs when a user does not follow proper C Shell syntax rules. For example, if a user types a command line with more left parentheses than right parentheses, a syntax error will occur.

Sequential Execution: A user may type multiple commands on the same command line. One way to do this is to use *sequential execution*, indicated by the semi-colon (";") character. For instance, if the user wants to know what the name of his current working directory is (using the *pwd* command), and get a list of the directory's contents (using the *ls* command), the user may type *pwd; ls*.

Pipes: Another way that users may type multiple commands on the same command line is to use *pipes*. A *pipe* is designated by the "|" character. A *pipe* allows a user to use the output from one command as the input to another command. For example, *who* is a program that lists all users that are currently logged on, and *grep* is a program that searches for particular string patterns. If I want to know if "myfriend" is logged on, I may type *who | grep myfriend*. If "myfriend" is logged on, "myfriend" and some other information output by the *who* program will be printed on my terminal screen. If "myfriend" is not logged on, then no information will be printed on my terminal screen.

Input/Output Redirection: Input/output redirection allows a user to direct output to a file that would otherwise have been printed on his terminal screen, and to take input from a file that would otherwise come from the terminal. Output redirection is designated by ">," and input redirection is designated by "<." For instance, *spell* is a program that prints misspelled words in a file on the user's terminal screen. If the user types *spell filename > errorfile*, *errorfile* will contain a list of all misspelled words in *filename*.

|| and &&: These are two relatively infrequently used features of the C Shell. If a user types *a || b*, then command "b" will be executed only if "a" fails. If a user types *a && b*, then command "b" will be executed only if "a" succeeds.

Appendix B

C Shell Trace Data Format

4 bytes: Unique user id
9 bytes: Beginning or Ending Time (System Clock)
1 byte: If a history substitution occurred, then this byte contains a 1.
If not, this byte contains a 0.
? bytes: Command line verbatim
(The verbatim command line is only recorded if history substitutions occurred.)
? bytes: Command line after history substitutions
1 byte: If aliasing occurred, then this byte contains a 1.
If not, then byte contains a 0.
? bytes: Command line after alias substitutions
(The command line after alias substitutions is only recorded if alias substitutions occurred.)
1 byte: Error and Source Code:
0 : No syntax error, command not executed from command file
1 : Syntax error, command not executed from command file
2 : No syntax error, command executed from command file, but not .login, .logout or .cshrc
3 : Syntax error, command executed from command file, but not .login, .logout or .cshrc
4 : No syntax error, command executed from .login
5 : Syntax error, command executed from .login
6 : No syntax error, command executed from .cshrc
7 : Syntax error, command executed from .cshrc
8 : No syntax error, command executed from .logout
9 : Syntax error, command executed from .logout

Notes

Error and Source Code: This byte indicates whether the command line contained a syntax error, and the source for the command. Commands may come from users typing at their terminal or various command files. The purposes of the three special command files, *.login*, *.cshrc*, and *.logout* are explained in Appendix A. Other command files may be used at any time.

Appendix C

Accounting Data Format

10 bytes: Command
2 bytes: User CPU time
2 bytes: System CPU time
2 bytes: Elapsed time
4 bytes: Beginning time (System Clock)
2 bytes: Unique user id
2 bytes: Accounting group id
2 bytes: Average memory usage
2 bytes: Number of disk I/O blocks
2 bytes: Tty
1 byte: Flag

Notes

CPU time is divided into system CPU time and user CPU time. System CPU time is CPU time spent executing system code for the user process, and user CPU time includes all other CPU time used by the user process. Average memory usage is recorded in 512 byte units. Each input/output operation corresponds to an input/output operation for one block. The size of the block is a multiple of 512 bytes, but no larger than 4096 bytes. (Each file has at most one block less than 4096 bytes in size.) The tty is an integer identifier assigned to each terminal. Dial-up lines, remote logins from other machines, and the system console are all assigned tty identifiers, along with "normal" terminals. The flag byte actually contains five flags. One flag is the fork flag indicating that the process has executed a fork() system call, but not an exec() system call. (See [Computer84a] for an explanation of fork() and exec().) The su flag indicates whether the process used super-user privileges. The compat flag indicates whether the process used compatibility mode. The core flag indicates whether the process dumped core, and the xsig flag indicates whether the process was killed by a signal.

Appendix D

Data Format after Combining Accounting Data and C Shell Trace Data

Data items that were added during the combine stage are indicated by asterisks.

** Byte 1 : Record Type
- 0 : Accounting File Record
  (Data from accounting file only)
- 1 : Combined Record
  (Data from both the accounting file and the C Shell trace file)
- 2 : C Shell only
  (Data from the C Shell trace file only)
- 3 : csh record from accounting file

** Byte 2-5 : Total number of bytes in record (excluding bytes 1,2,3,4,5)
** Byte 6 : Machine id

Accounting Record or csh Record from Accounting File:
- Bytes 7-8 : Unique user id
- ** Bytes 9-16 : Login ID of User
- Bytes 17-20 : Beginning time (System Clock)
- Bytes 21-30 : Command
- Bytes 31-32 : System CPU time
- Bytes 33-34 : User CPU time
- Bytes 35-36 : Elapsed time
- Bytes 37-38 : Average memory usage
- Bytes 39-40 : Number of disk I/O blocks
- Bytes 41-42 : tty
- Byte 43 : Flag

Combined Record or C Shell Only Record:
- Bytes 7-10 : Beginning or Ending time (System Clock)
- Bytes 11-12 : Unique user id
- Bytes 13-20 : Login ID of User
- ** Bytes 21-22 : Length of next field
- Bytes 23-? Command line verbatim
- Next byte Error and Source code:
  - 0 : No syntax error, command not executed from command file
  - 1 : Syntax error, command not executed from command file
  - 2 : No syntax error, command executed from command file, but not .login, .logout or .cshrc
  - 3 : Syntax error, command executed from command file, but not .login, .logout or .cshrc
  - 4 : No syntax error, command executed from .login
  - 5 : Syntax error, command executed from .login
  - 6 : No syntax error, command executed from .cshrc

7 : Syntax error, command executed from .cshrc
8 : No syntax error, command executed from .logout
9 : Syntax error, command executed from .logout

** Next 2 bytes Length of next field
?-? Command line with history substitutions
** Next 2 bytes Length of next field
?-? Command line with history substitutions and aliases replaced
** Next byte Accounting count : Number of accounting and multiple-accounting sub-records to follow
Note: Multiple account records for one command are only counted once.
** Next byte Built-in count : Number of sub-records to follow which correspond to commands built-in to the C-Shell
** Next byte Missing count : Number of sub-records to follow which correspond to commands which should have had accounting records, but did not
(NOTE: The order of the accounting, built-in, and missing records corresponds to the order of the commands in the command line.)

Accounting sub-record:
** One byte: 0
Same as bytes 17-43 in accounting record.

Multi-Accounting sub-record:
** One byte: 3
** 10 bytes: Command from trace record
** Two bytes: Number of account records to follow
Same as bytes 17-43 in accounting record.

Built-in sub-record:
** One byte: 1 (Built-in sub-record id)
** 10 bytes: Command from trace record

Missing sub-record:
** One byte: 2 (Missing sub-record id)
** 10 bytes: Command from trace record

Notes

Machine and User Identifiers: A unique identifier was assigned to the machine and each user. All my data is from one machine, but I decided to record a machine identifier, in case data was later collected from other machines. The system assigns to each user both an integer user identifier and a character string identifier, which is his login identifier. The user must type his login identifier whenever he logs in. The user is usually unaware that he is also assigned an integer login identifier, as this is used only by the system.

Appendix E

Data Format after Post-Processing

Data items that were added during post-processing are indicated by asterisks.

Byte 1 : Record Type
- 0 : Accounting File Record
  (Data from the accounting file only)
- 1 : Combined Record
  (Data from both the accounting file and the C Shell trace data file.)
- 2 : C Shell only
  (Data from the C Shell trace data file only.)
- 3 : csh record from accounting file

Byte 2-5 : Total number of bytes in record (excluding bytes 1,2,3,4,5)
Byte 6 : Machine id
** Bytes 7-10 : New user id's. (Small integers)
** Byte 11: User classification info - Sponsor
** Byte 12: User classification info - User classification (type of work)
** Bytes 13-16: Login session id

Accounting Record or csh Record from accounting file: .
- 2 bytes : Unique system user id
- 8 bytes : Login ID of User
- 4 bytes : Beginning time (System Clock)
- ** 4 bytes : Command id number
- ** 1 byte : Command category
- 10 bytes : Command
- 2 bytes : System CPU time
- 2 bytes : User CPU time
- 2 bytes : Elapsed time
- 2 bytes : Average memory usage
- 2 bytes : Number of disk I/O blocks
- 2 bytes : tty
- ** 1 byte : Terminal classification: remote login, hardwired, patchboard, dialup, console, or other
- 1 byte : Flag

Combined Record or C Shell Record:
- 4 bytes : Beginning or Ending time (System Clock)
- 2 bytes : Unique system user id
- 8 bytes : Login ID of User
- ** 1 byte : Flag to indicate background or foreground command
- 2 bytes : Length of next field
- ? bytes : Command line verbatim
- Next byte Error and Source code:
  - 0 : No syntax error, command not executed from command file
  - 1 : Syntax error, command not executed from command

file
2 : No syntax error, command executed from command file, but not .login or .cshrc
3 : Syntax error, command executed from command file, but not .login or .cshrc
4 : No syntax error, command executed from .login
5 : Syntax error, command executed from .login
6 : No syntax error, command executed from .cshrc
7 : Syntax error, command executed from .cshrc
8 : No syntax error, command executed from .logout
9 : Syntax error, command executed from .logout

Next 2 bytes Length of next field
?-? Command line with history substitutions
Next 2 bytes Length of next field
?-? Command line with history substitutions and aliases replaced
Next byte Accounting count : Number of accounting sub-records to follow
Next byte Built-in count : Number of sub-records to follow which correspond to commands built-in to C-Shell
Next byte Missing count : Number of sub-records to follow which correspond to command which should have had accounting records, but did not
(NOTE: The order of the accounting, built-in, and missing records corresponds to the order of the commands in the command line.)

Accounting sub-record:
One byte: 0
Same as (beginning time to end of record) in accounting record.

Multi-Accounting sub-record:
One byte: 4
** 4 bytes: Command id number for command in trace record
** 1 byte: Command category for command in trace record
10 bytes: Command from trace record
Two bytes: Number of acct records to follow
Same as (beginning time to end of record) in accounting record.

Built-in sub_record:
One byte: 1 (Built-in sub-record id)
** 4 bytes: Command id number
** 1 byte : Command category
10 bytes: Command from trace record

Missing sub-record:
One byte: 2 (Missing sub-record id)
** 4 bytes: Command id number
** 1 byte : Command category
10 bytes: Command from trace record

Appendix F

Table 29 - Number of Command Lines						
	Terminal	.login	.cshrc	.logout	Command File	Total
	9316	3804	10223	564	6605	30512
	2.1%	0.8%	2.3%	0.1%	1.5%	6.7%
Secretary	30.5%	12.5%	33.5%	1.8%	21.6%	100.0%
	12985	3810	15712	449	26423	59379
	2.9%	0.8%	3.5%	0.1%	5.8%	13.1%
Facu	21.9%	6.4%	26.5%	0.8%	44.5%	100.0%
	24541	19737	37614	1639	10207	93738
	5.4%	4.4%	8.3%	0.4%	2.3%	20.7%
Grad	26.2%	21.1%	40.1%	1.7%	10.9%	100.0%
	13939	8636	14478	693	2302	40048
	3.1%	1.9%	3.2%	0.2%	0.5%	8.8%
Guest	34.8%	21.6%	36.2%	1.7%	5.7%	100.0%
	10832	5618	12282	149	6667	35548
	2.4%	1.2%	2.7%	0.0%	1.5%	7.8%
Misc-Human	30.5%	15.8%	34.6%	0.4%	18.8%	100.0%
	1	—	2	—	130675	130678
	0.0%	—	0.0%	—	28.8%	28.8%
Misc-Non-Human	0.0%	—	0.0%	—	100.0%	100.0%
	11232	11633	15290	800	13075	52030
	2.5%	2.6%	3.4%	0.2%	2.9%	11.5%
Prog	21.6%	22.4%	29.4%	1.5%	25.1%	100.0%
	2265	—	4731	—	4335	11331
	0.5%	—	1.0%	—	1.0%	2.5%
Root	20.0%	—	41.8%	—	38.3%	100.0%
	—	—	—	—	—	—
	—	—	—	—	—	—
Unknown	—	—	—	—	—	—
	85111	53238	110332	4294	200289	453264
	18.8%	11.7%	24.3%	0.9%	44.2%	100.0%
Totals	18.8%	11.7%	24.3%	0.9%	44.2%	100.0%

Appendix G

Table 30 - Average Length of User's Command Line						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	8	38	28	12	24	22
Facu	8	55	22	9	22	21
Grad	7	37	24	15	22	22
Guest	7	37	30	23	23	23
Misc-Human	8	43	26	15	24	23
Misc-Non-Human	28	—	120	—	23	23
Prog	7	33	22	12	16	20
Root	9	—	28	—	18	21
Unknown	—	—	—	—	—	—
Overall	7	38	25	15	22	22

The above table shows the average length of all command lines, before alias and history substitutions were made.

Table 31 - History User's Average Command Line Length						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	5	—	—	27	—	6
Facu	4	—	—	—	—	4
Grad	4	—	—	—	—	4
Guest	6	—	—	—	—	6
Misc-Human	4	—	—	—	—	4
Misc-Non-Human	—	—	—	—	—	—
Prog	5	—	—	—	—	5
Root	6	—	—	—	—	6
Unknown	—	—	—	—	—	—
Overall	4	—	—	27	—	4

The above table shows the average length of a command line that used history substitutions, BEFORE the history substitutions were made.

Table 32 - Alias User's Average Command Line Length						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	6	36	22	17	—	9
Facu	7	14	22	2	20	17
Grad	5	25	22	7	24	16
Guest	4	30	31	6	32	21
Misc-Human	6	21	19	—	22	16
Misc-Non-Human	—	—	—	—	—	—
Prog	4	32	19	3	20	12
Root	7	—	—	—	18	7
Unknown	—	—	—	—	—	—
Overall	5	24	22	7	22	16

The above table shows the average length of a command line that used alias substitutions, BEFORE the alias substitutions were made. If the command line used both history and alias substitutions, the command line length is the command line length before either history or alias substitutions were made.

Table 33 - Average History Command Line Length						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	16	—	—	30	—	16
Facu	15	—	—	—	—	15
Grad	14	—	—	—	—	14
Guest	17	—	—	—	—	17
Misc-Human	13	—	—	—	—	13
Misc-Non-Human	—	—	—	—	—	—
Prog	15	—	—	—	—	15
Root	19	—	—	—	—	19
Unknown	—	—	—	—	—	—
Overall	15	—	—	30	—	15

The above table shows the average length of a command line that used history substitutions. AFTER the history substitutions were made.

Table 34 - Average Alias Command Line Length						
	Terminal	.login	.cshrc	.logout	Command File	Overall
Secretary	11	60	27	20	—	15
Facu	14	69	26	6	27	22
Grad	13	35	26	14	30	22
Guest	16	49	35	15	40	29
Misc-Human	12	26	23	—	26	21
Misc-Non-Human	—	—	—	—	—	—
Prog	12	81	23	6	24	18
Root	16	—	—	—	36	16
Unknown	—	—	—	—	—	—
Overall	13	34	26	14	27	22

The above table shows the average length of a command line that used alias substitutions, AFTER the alias substitutions were made. If the command line used both history and alias substitutions, the command line length is the command line length after both history and alias substitutions were made.